USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Determinación do camiño seguro para a neutralización dun incidente químico

Eva Miranda Barreiro

2018/2019

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Determinación do camiño seguro para a neutralización dun incidente químico

Eva Miranda Barreiro

Xullo de 2019

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Traballo proposto

Área de Coñecemento: Análise Matemática
Título: Determinación do camiño seguro para a neutralización dun incidente químico
Breve descrición do contido
Trátase de analizar matematicamente e resolver numericamente un problema relacionado coa busca do mellor camiño que nos permita neutralizar un incidente químico de tipo industrial (TIC) ou doutra clase. O obxectivo fundamental será atopar un camiño que nos permita chegar o máis próximo posible ó punto de vertido, de forma que a concentración total ó longo de dito camiño estea por debaixo dun límite tolerable.
Recomendacións
Lectura recomendada: J. M. Stockie, «The Mathematics of Atmosferic Dispersion Modeling,» SIAM (Society for Industrial and Applied Mathematics) Review , vol. 53, nº 2, pp. 349-372, 2011
Outras observacións

Índice xeral

Resumo

Neste traballo modélase o problema de búsqueda dunha ruta segura no proceso de neutralización dun incidente químico. Mediante a resolución de ecuacións en derivadas parciais obteremos o modelo de dispersión *Gaussian Plume*, co que daremos forma ao problema en cuestión. Faremos unha resolución numérica dun exemplo concreto de interese utilizando a ferramenta MATLAB e expoñeremos as conclusións extraídas durante todo o proceso.

Abstract

In this paper we model the problem of finding a safe route in the process of neutralizing a chemical incident. By solving equations in partial derivatives we obtain the Gaussian Plume dispersion model, with which we will give shape to the problem in question. We will make a numerical resolution of a concrete example of interest using MATLAB and we will expose the conclusions drawn during the whole process.

Introdución

Os axentes químicos como tal tiveron un empuxe no seu uso industrial a finais do século XIX, e ata 70 produtos químicos foron utilizados como armas durante o século XX. Desde este período tiveron lugar varios episodios de desastres industriais por incidentes químicos, como o escape de gas cloro en Romanía no 1939 ou o escape de metil isocianato en Bhopal (India) que xerou unha nube tóxica sobre a cidade no 1984. Ambos acontecementos causaron numerosas mortes entre a poboación.

Motivados por este tipo de incidentes, a comunidade científica centrouse en definir modelos de dispersión que describiran con precisión a concentración de contaminante por zona, tendo en conta as condicións de entorno nas que se atopa a área afectada.
Un problema resolto grazas a estes modelos é a determinación do ratio de acción dun punto de emisión (ou varios) baseándonos nun conxunto de condicións do entorno. Este tipo de proxectos permite realizar un control de emisións a nivel industrial.

Neste traballo faremos uso deses modelos de dispersión, máis concretamente do modelo *Gaussian Plume*, para resolver o problema de determinación do camiño seguro para a neutralización dun incidente químico.

En España este tipo de problema está recollido no PEE (Plan de Emerxencias Exteriores) [2]. Aínda que sempre se deseña un plan de prevención para esta clase de incidentes, non sempre son eficaces e incluso en ocasións é necesaria unha intervención directa. Nese caso é o grupo ESI (Equipo de Segunda Intervención) o encargado de facerse cargo da situación, que non son máis que os bombeiros cos equipos e medios adecuados. O problema principal redúcese entón a evitar a contaminación do persoal do servizo de emerxencia, ou en todo caso minimizalo. Por iso debe facerse un estudo do entorno e ter en conta todas as condicións climáticas.

Capítulo 1

Descrición do problema

Neste primeiro capítulo buscamos obter unha ecuación que modele a dispersión dun contaminante na atmosfera para un tipo de incidente químico concreto. Nos seguintes capítulos faremos as simplificacións pertinentes para chegar a un modelo de dispersión concreto.

Consideremos pois a *lei de conservación da masa* asociada á contaminación,

$$\frac{\partial C}{\partial t} + \nabla \cdot \overrightarrow{J} = S. \tag{1.1}$$

C: concentración de masa do contaminante, $C(\vec{x}, t)$ $[kg/m^3]$ para un $\overrightarrow{x} = (x, y, z) \in \mathbb{R}^3$ $[m]$ y $t \geq 0$ $[s]$, $t \in \mathbb{R}$.

S: término fonte, $S(\vec{x}, t)$ $[\mathrm{kg/m^3/s}]$.

J: fluxo de masa do contaminante debido, principalmente, os efectos da advección e da difusión, $\overrightarrow{J} = (\overrightarrow{x}, t)[kg/m^3\, s]$.

1.1. Difusión

Definimos difusión como o proceso de propagación das moléculas do compoñente químico en cuestión a través dun medio que antes estaba libre de contaminación. Verase representada mediante o termo D. Para obter o coeficiente de difusión aplicaremos a coñecida como $Lei\, de\, Fick$,

$$\overrightarrow{J}_D = -D\frac{\partial C}{\partial x}. \tag{1.2}$$

Asumimos que a difusión do fluxo é proporcional ao gradiente da concentración. Como

nos atopamos en $\mathbb{R}^3$ a Lei fe Fick (1.2) ten a seguinte expresión:

$$\overrightarrow{J}_D = -D\nabla C, \tag{1.3}$$

onde $D(\overrightarrow{x})\,[m^2/s]$ se corresponde coa matriz de difusións

$$D(\overrightarrow{x}) = \begin{pmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{pmatrix}.$$

1.2. Advección

O proceso de advección consiste no transporte das partículas de contaminante a través do medio. No noso caso este proceso vese realizado polo vento, que produce unha aportación ao fluxo de forma lineal, a cal podemos expresar por

$$\overrightarrow{J}_A = C\overrightarrow{u}, \tag{1.4}$$

sendo $\overrightarrow{u} \in \mathbb{R}^3$ o vector velocidade do vento $[m/s]$.

1.3. Ecuacións principais

Polo tanto, se temos en conta o anterior e xuntamos (1.3) e (1.4), obtemos que:

$$\overrightarrow{J} = \overrightarrow{J}_D + \overrightarrow{J}_A = -D\nabla C + C\overrightarrow{u}. \tag{1.5}$$

Substituíndo na ecuación (1.1) pola nova expresión de J, (1.5), temos que:

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\overrightarrow{u}) = \nabla \cdot (D\nabla C) + S, \tag{1.6}$$

coñecida como a **Ecuación de Advección-Difusión.** Se queremos detallar un pouco máis esta ecuación, tomando

$$\nabla \cdot \overrightarrow{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}, \quad \forall\, \overrightarrow{v} \in \mathbb{R}^3, \tag{1.7}$$

(definición de diverxencia para un vector) entón,

$$\nabla \cdot (C\overrightarrow{u}) = \nabla C \cdot \overrightarrow{u} + C\nabla \cdot \overrightarrow{u} = \frac{\partial C}{\partial x}u_x + \frac{\partial C}{\partial y}u_y + \frac{\partial C}{\partial z}u_z. \tag{1.8}$$

A última igualdade é consecuencia de supoñer que $C\nabla \cdot \overrightarrow{u} = 0$, ao considerar que estamos en niveles baixos da atmosfera (supoñemos que se comporta como un fluido incompresible). Por outra parte:

$$\nabla \cdot (D\nabla C) = \frac{\partial}{\partial x}(D_x\frac{\partial C}{\partial x}) + \frac{\partial}{\partial y}(D_y\frac{\partial C}{\partial y}) + \frac{\partial}{\partial z}(D_z\frac{\partial C}{\partial z}) \tag{1.9}$$

Reescribimos pois a ecuación (1.6)

$$\frac{\partial C}{\partial t} + \frac{\partial C}{\partial x}u_x + \frac{\partial C}{\partial y}u_y + \frac{\partial C}{\partial z}u_z = \frac{\partial}{\partial x}(D_x\frac{\partial C}{\partial x}) + \frac{\partial}{\partial y}(D_y\frac{\partial C}{\partial y}) + \frac{\partial}{\partial z}(D_z\frac{\partial C}{\partial z}) + S \quad (1.10)$$

Esta será a ecuación principal que expresa a evolución da concentración da contaminación nun medio. Pasará por uns procesos de simplificación e plantearemos con ela un problema en derivadas parciais máis sinxelo, do cal obteremos unha solución concreta que resultará ser o coñecido como Modelo *Gaussian Plume*.

Capítulo 2

Análise matemática

Neste capítulo damos as bases para obter de forma simple o Modelo *Gaussian Plume* partindo do traballo realizado por O.F.T. Roberts [11] e O.G. Sutton [17]. A finalidade é atopar, unha solución particular para os problemas de dispersión atmosférica supoñendo un conxunto de hipóteses que simplifican o proceso de resolución: condicións de fronteira e dependencia de parámetros.

2.1. Modelo *Gaussian Plume*

O modelo *Gaussian Plume* toma o seu nome da forma que adquire o contaminante no seu proceso de dispersión, en forma de "penacho"(*plume*) e seguindo unha distribución *Gaussiana*, como podemos observar na figura 2.1.

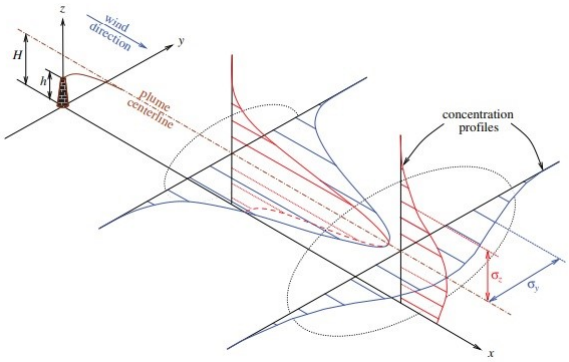


Figura 2.1: Representación gráfica do Modelo *Gaussian Plume*. (Contido extraído do artigo *The Mathematics of Atmospheric Dispersion Modeling* [16]).

Este modelo é a solución máis simple dos problemas de dispersión atmosférica cando

tomamos certas condicións iniciais e de fronteira. Considérase como a forma máis estándar do estudio do transporte de partículas contaminantes a través dun medio tendo en conta os efectos de difusión e advección. Correspóndese con un punto fonte que emite un contaminante de forma unidireccional, a través do vento e nun dominio de extensión infinita. Algunha das aplicacións máis comúns de este modelo danse nos estudios de emisións que se producen en procesos industrais de gran envergadura, aínda que tamén ten utilidade para a diseminación das cinzas volcánicas debido a unha erupción ou na propagación do cheiro que provén de instalacións gandeiras.

A continuación imos ter en conta unha serie de suposicións para poder simplificar a ecuación (1.10) e que nos permitan chegar a algunha solución particular.

1. Definición do término fonte

   Consideramos que a substancia contaminante se emite baixo unha tasa constante $Q\,[\mathrm{kg\,s^{-1}}]$ desde un único punto fonte $x = (0, 0, H)$, a unha altura H sobre a superficie. Polo tanto, consideraremos o seguinte termo fonte:

   $$S = Q(x)\delta(y)\delta(z - H). \tag{2.1}$$

   onde $\delta(x)$ é a distribución (dual do espazo das funcións clase infinito con soporte compacto) Delta de Dirac centrada no cero, que podemos definir formalmente como:

   $$\delta(x) = \begin{cases} \infty, & x = 0, \\ 0, & x \neq 0. \end{cases} \tag{2.2}$$

   Por resultados clásicos dos espazos de distribucións (ver [13]), sabemos que a anterior distribución pode ser aproximada por unha sucesión regularizante. Dita aproximación, tal e como veremos máis adiante, será fundamental para obter a solución de forma explícita.

2. Velocidade do vento constante e paralela ó eixo OX. Tomamos un vector velocidade $\overrightarrow{u}$ constante e na dirección do eixo OX.

   $$\overrightarrow{u} = (u, 0, 0),\ u \geq 0.$$

3. Solución estacionaria. Suporemos que a concentración non varía co tempo:

   $$\frac{\partial C}{\partial t} = 0. \tag{2.3}$$

4. Difusión constante en todas as direccións. Imos considerar que as constantes de difusión son iguais en todos os eixos, e que dependen unicamente de x: $K_x = K_y =$

$K_z = K(x)$, con K unha función continua en x. Terase entón que:

$$\begin{aligned} \nabla \cdot (K \nabla C) &= \frac{\partial}{\partial x}(K_x \frac{\partial C}{\partial x}) + \frac{\partial}{\partial y}(K_x \frac{\partial C}{\partial y}) + \frac{\partial}{\partial z}(K_x \frac{\partial C}{\partial z}) \\ &= \frac{\partial}{\partial x}(K_x \frac{\partial C}{\partial x}) + K_x \frac{\partial^2 C}{\partial y^2} + K_x \frac{\partial^2 C}{\partial z^2}. \end{aligned} \tag{2.4}$$

5. Convección dominante no eixo OX. Desprezamos os efectos difusores no eixo OX:

$$\frac{\partial}{\partial x}(K_x \frac{\partial C}{\partial x}) \cong 0. \tag{2.5}$$

6. Dominio semi-infinito e chan. Imos despreciar tamén as variacións da topografía e considerar que o chan atópase en $z = 0$, polo tanto estamos tomando:

$$\Omega = [0, \infty] \times (-\infty, \infty) \times [0, \infty).$$

7. Impermeabilidade do chan. Imos considerar que o contaminante non penetra no chan,

$$K \frac{\partial C}{\partial z}(x, y, 0) = 0. \tag{2.6}$$

Unha vez consideradas as anteriores simplificacións, podemos reescribir (1.10) da seguinte forma:

$$u \frac{\partial C}{\partial x} = K_x \frac{\partial^2 C}{\partial y^2} + K_x \frac{\partial^2 C}{\partial z^2} + Q(x)\delta(y)\delta(z - H), \tag{2.7}$$

xunto coas seguintes condicións de fronteira e iniciais:

$$\begin{aligned} C(0, y, z) &= 0, \\ C(\infty, y, z) &= 0, \\ C(x, \infty, z) &= 0, \\ C(x, -\infty, z) &= 0, \\ C(x, y, \infty) &= 0, \\ K \frac{\partial C}{\partial z}(x, y, 0) &= 0. \end{aligned} \tag{2.8}$$

Unha formulación equivalente de este problema ven dada por

$$u \frac{\partial C}{\partial x} = K_x \frac{\partial^2 C}{\partial y^2} + K_x \frac{\partial^2 C}{\partial z^2}, \tag{2.9}$$

onde pasamos o término fonte ás condicións de fronteira:

$$\begin{aligned} C(0, y, z) &= Q(x)\delta(y)\delta(z - H), \\ C(\infty, y, z) &= 0, \\ C(x, \infty, z) &= 0, \\ C(x, -\infty, z) &= 0, \\ C(x, y, \infty) &= 0, \\ K \frac{\partial C}{\partial z}(x, y, 0) &= 0. \end{aligned} \tag{2.10}$$

As equivalencias entre os problemas (2.7) e (2.9) veñen detalladas en [15]. A idea principal da que parte esta equivalencia é da relación existente entre a *Ecuación de Conducción da Calor*

$$\begin{aligned} &\frac{\partial u(x,t)}{\partial t} - a\nabla^2 u = 0, \quad t > 0, \\ &u(x,0^+) = \delta(x), \end{aligned} \tag{2.11}$$

e da solución do problema

$$\begin{aligned} &\frac{\partial C(x,t)}{\partial t} - a\nabla^2 C = \delta(x)\delta(t), \\ &C \equiv 0, \quad t < 0, \end{aligned} \tag{2.12}$$

que ven dada por $C(x,t) = u_0(t)u(x,t)$, sendo $u_0(t)$ a función de *Heaviside* que trataremos na sección seguinte.

2.2. A transformada de Laplace e as súas propiedades

Posto que empregaremos a Transformada de Laplace para resolver a ecuación (2.7), introduciremos nesta sección a súa definición e propiedades máis importantes. Ningún resultado desta sección conta con demostración, para maior comprobación acudir á referencia [12].

Definición 2.1 (Transformada de Laplace)**.** Sexa $f : A \subset \mathbb{R} \longrightarrow \mathbb{R}$, para $t > 0$ con $s \in \mathbb{R}$, definimos a Transformada de Laplace de f:

$$\begin{aligned} F(s) = \mathcal{L}(f(t)) &= \int_0^\infty e^{-st} f(t)dt \\ &= \lim_{r\to\infty} \int_0^\tau e^{-st} f(t)dt, \end{aligned} \tag{2.13}$$

cando o límite existe e é finito.

Se non existe este límite, por ser a integral diverxente, non podemos definir a transformada de Laplace. Unha das propiedades máis importantes da transformada de Laplace é a linealidade, isto é, se f e g son dúas funcións para as cales podemos definir a súa transformada de Laplace, terase que:

- $\mathcal{L}(f(t)) + \mathcal{L}(g(t)) = \mathcal{L}(f(t) + g(t))$,
- $\lambda\mathcal{L}(f(t)) = \mathcal{L}(\lambda\, f(t))$.

Ademais desta propiedade, veremos outras que nos van permitir resolver problemas de valor inicial asociados a ecuacións diferenciais ordinarias e, incluso, ecuacións en derivadas parciais como é o caso que nos ocupa.

Definición 2.2. Unha función f dicimos que é de orden expoñencial α se existen unhas constantes $M > 0$ e α de tal forma que para algún $t_0 \geq 0$,

$$|f(t)| \leq M\, e^{\alpha t}, \qquad t \geq t_0. \tag{2.14}$$

Teorema 2.3. *Si f é unha función continua definida a trozos en $[0, \infty)$ e de orden expoñencial α, entón a transformada de Laplace $\mathcal{L}(f)$ existe para $s > \alpha$.*

Definición 2.4. Se $\mathcal{L}(f(t)) = F(s)$ definimos a inversa da transformada de Laplace como:

$$\mathcal{L}^{-1}(F(s)) = f(t), \quad t \geq 0, \tag{2.15}$$

que permite obter a partir da transformada a función orixinal da que provén.

Teorema 2.5 (de Lerch)**.** *Sexan f e g dúas funcións continuas a trozos de orden expoñencial α en $[0, \infty)$ e supoñamos que $\mathcal{L}(f(t)) = \mathcal{L}(g(t))$. Entón $f(t) = g(t)$ en todo punto t no que ambas funcións son continuas.*

A inversa da transformada de Laplace tamén verifica a propiedade de linealidade:

$$\mathcal{L}^{-1}(aF(s) + bG(s)) = af(t) + bg(t), \tag{2.16}$$

con $\mathcal{L}(f(t)) = F(s)$ e $\mathcal{L}(g(t)) = G(s)$.

Exemplo 2.6. Definimos a función *Heaviside*, $u_a(t)$, tal que:

$$u_a(t) = \begin{cases} 1 & t \geq a \\ 0 & t < a \end{cases} \tag{2.17}$$

ou

$$u_a(t) = \begin{cases} 1 & t > a \\ 0 & t < a \end{cases} \tag{2.18}$$

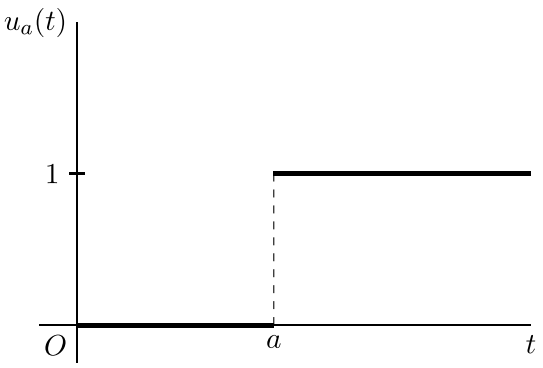


Figura 2.2: Representación gráfica da función Heaviside

Ambas definicións para $u_a(t)$ posúen a mesma transformada de Laplace,

$$\mathcal{L}(u_a(t)) = \frac{e^{-as}}{s}. \tag{2.19}$$

Se consideramos $a = 0$,

$$u_a(t) = u(t) \begin{cases} 1 & t > 0 \\ 0 & t < 0 \end{cases}, \tag{2.20}$$

e podemos observar que $u_a(t) = u(t-a)$.

Os seguintes resultados son moi útiles para o proceso de obtención da transformada e da súa inversa.

Teorema 2.7 (Primeiro Teorema de Traslación)**.** *Se $F(s) = \mathcal{L}(f(t))$ para $s > 0$, entón*

$$F(s-a) = \mathcal{L}\left(e^{at} f(t)\right), \quad a \text{ real}, \mathcal{R}e(s) > a.$$

Teorema 2.8 (Segundo Teorema de Traslación)**.** *Se $F(s) = \mathcal{L}(f(t))$, entón*

$$\mathcal{L}\left(u_a(t) f(t-a)\right) = e^{-as} F(s), \qquad a \geq 0.$$

Este resultado tamén pode ser considerado na forma inversa:

$$\mathcal{L}^{-1}\left(e^{-as} F(s)\right) = u_a(t) f(t-a),$$

para $F(s) = \mathcal{L}(f(t))$, $a \geq 0$.

A transformada de Laplace permite resolver ecuacións diferenciais dunha forma máis sinxela, procedemos pois a definir a transformada de Laplace da función derivada de f, $\mathcal{L}(f'(t))$, en términos de $\mathcal{L}(f(t))$.

Teorema 2.9 (Teorema de derivación)**.** *Supoñemos f continua en $(0, \infty)$ y de orden exponencial α, e que f' é continua a trozos en $[0, \infty)$, entón:*

$$\mathcal{L}(f'(t)) = s\mathcal{L}(f(t)) - f(0^+) \qquad s > \alpha.$$

Observación 2.10. Se f é unha función continua temos que $f(0) = f(0^+)$.

Teorema 2.11. *Supoñemos que f é continua en $[0, \infty)$, excepto en $t = t_1 > 0$ onde ten unha discontinuidade de salto. Supoñemos tamén f de orden expoñencial α, con f' continua a trozos en $[0, \infty)$, entón:*

$$\mathcal{L}(f'(t)) = s\mathcal{L}(f(t)) - f(0) - e^{t_1 s}\left(f(t_1^+ - f(t_1^-)\right).$$

Teorema 2.12. *Supoñamos que f(t), $f'(t), \cdots, f^{n-1)}(t)$ son continuas en $(0,\infty)$ e de orden expoñencial α, mentres que $f^{n)}(t)$ é continua a trozos en $[0,\infty)$, entón:*

$$\mathcal{L}\left(f^{n)}(t)\right) = s^n\mathcal{L}(f(t)) - s^{n-1}f\left(0^+\right) - s^{n-2}f'\left(0^+\right) - \cdots - f^{n-1)}\left(0^+\right).$$

A continuación, damos un esquema básico de un método para resolver EDOs ou EDPs coa transformada de Laplace:

1. Tomamos a transformada a ambos lados da ecuación e obtemos o que podemos considerar como *ecuación transformada.*
2. Obtemos unha ecuación $\mathcal{L}(y) = F(s)$ onde F é unha expresión alxébrica que depende de s.
3. Aplicamos o proceso de transformada inversa, $y = \mathcal{L}^{-1}(F(s))$.

2.3. A Delta de Dirac

Para definir a Delta de Dirac, δ, necesitamos considerar algunhas definicións e resultados relacionados coa teoría das distribucións.

Definición 2.13 (Soporte dunha función)**.** Definimos como soporte dunha función ao conxunto de puntos onde a función non se anula, $A = \{x \,/\, f(x) = 0\}$, ou á clausura dese conxunto $\overline{A}$.

Dicimos que unha función é de soporte compacto se $\overline{A}$ é un conxunto pechado e acotado.

Definición 2.14 (Funcións de clase infinito con soporte compacto)**.** Denotaremos por $\mathcal{D}$ ao espacio vectorial formado polas funcións φ en n variables reais, de clase infinito e con soporte compacto.

Definición 2.15 (Distribución)**.** Unha distribución T é unha forma lineal continua sobre $\mathcal{D}$. Denotaremos ó espazo das distribucións como $\mathcal{D}'$. Dada unha distribución $T \in \mathcal{D}'$ e un elemento $\varphi \in \mathcal{D}$, existen notacións distintas para expresar unha distribución:

$$\varphi \longrightarrow T(\varphi) \text{ ó } T\,\mathrm{o}\,\varphi \text{ ó } < T, \varphi > .$$

Exemplo 2.16. Dada unha función localmente integrable (integrable en calquera compacto) $f \in L^1_{loc}(\mathbb{R})$, terase que:

$$\begin{array}{rcl} T_f : \mathcal{D} & \longrightarrow & \mathbb{R} \\ \varphi & \longrightarrow & < T_f, \varphi >= \displaystyle\int_{\mathbb{R}^n} f\,\varphi\,dx \end{array} \tag{2.21}$$

é unha distribución.

Observación 2.17. Tanto o espazo $\mathcal{D}$, como o espazo $\mathcal{D}'$, pódense dotar dunha topoloxía. Neste traballo non abordaremos a súa construción, empregaremos eso si, a propiedade que nos garante que unha sucesión de distribucións $\{T_n\}_{n\in\mathbb{N}}$ é converxente a $T \in \mathcal{D}'$ se e soamente se:

$$< T_n, \varphi > \longrightarrow < T, \varphi >, \quad \forall \varphi \in D.$$

Exemplo 2.18 (Delta de Dirac)**.** Definimos a Delta de Dirac, δ, como a seguinte distribución:

$$\begin{array}{rccl} \delta: & \mathcal{D} & \longrightarrow & \mathbb{R} \\ & \varphi & \longrightarrow & < \delta, \varphi >= \varphi(0). \end{array} \tag{2.22}$$

É dicir, a Delta de Dirac é unha distribución que devolve o valor da función φ na orixe. Outra forma de obter a Delta de Dirac é vendo que é a derivada no sentido das distribucións (se $T \in \mathcal{D}'$, entón, $< T', \varphi >= (-1) < T, \varphi' >$, $\forall \varphi \in \mathcal{D}$) da función de *Heaviside*, (2.18), para $a = 0$:

$$Y(t) = u_0(t) = \begin{cases} 1 & t > 0 \\ 0 & t < 0. \end{cases} \tag{2.23}$$

Agora ben, tomando $Y(\varphi) = \displaystyle\int_0^\infty \varphi(x)\, dx$,

$$Y'(\varphi) = -Y(\varphi') = -\int_0^\infty \varphi'(x)\, dx = \varphi(0) = \delta(\varphi). \tag{2.24}$$

Temos que, $Y' = \delta$. Podemos empregar este feito xunto coa relación que existe para a transformada de Laplace da derivada, para obter, formalmente (non definimos a transformada de Laplace para distribucións), a transformada de Laplace da Delta de Dirac. En efecto:

$$\mathcal{L}(\delta(t - t_0)) = \int_0^\infty e^{-st} \cdot \delta(t - t_0) = e^{-st_0}, \quad \forall\, t > 0. \tag{2.25}$$

Por definición da Delta de Dirac, (2.24), temos que

$$\delta(t - t_0) = \frac{d}{dt} Y(t - t_0), \tag{2.26}$$

entón,

$$\mathcal{L}(\delta(t - t_0)) = \int_0^\infty e^{-st} \cdot \frac{d}{dt} Y(t - t_0). \tag{2.27}$$

Realizando o proceso de integración por partes para $u = e^{-st}$ e $dv = \frac{d}{dt} Y(t - t_0)\, dt$ temos

$$\begin{aligned} \mathcal{L}(\delta(t - t_0)) &= e^{-st} \cdot Y(t - t_0)\big]_{t_0}^{\infty} - \int_{t_0}^{\infty} Y(t - t_0) \cdot \left(-e^{-st}\right) dt \\ &= \int_{t_0}^{\infty} e^{-st}\, dt \\ &= e^{-st_0}, \quad \forall\, t > 0. \end{aligned} \tag{2.28}$$

No seguiente teorema veremos que a Delta de Dirac pode ser aproximada por unha sucesión de funcións integrables, sempre e cando cumpran unha serie de condicións.

Teorema 2.19 (Aproximación da Delta de Dirac)**.** *Sexa* $\{h_n\}_{n\in\mathbb{N}} \subset L^1(\mathbb{R})$ *tal que,*

1. $h_n(x) \geq 0$, $\forall n \in \mathbb{N}$,
2. $\displaystyle\int_{\mathbb{R}} h_n(x)\,dx = 1, \quad \forall n \in \mathbb{N}$,
3. $\displaystyle\lim_{n\to\infty} \int_{\mathbb{R}} |x|\, h_n(x)\,dx = 0$,

entón:

$$\lim_{n\to\infty} \int_{\mathbb{R}} h_n = \delta_0, \quad en\ \mathcal{D}'. \tag{2.29}$$

Demostración. Por unha banda, tense que o espazo $L^1(\mathbb{R}) \subset L^1_{loc}(\mathbb{R})$, polo tanto, calquer función de $L^1(\mathbb{R})$ pode ser entendida como unha distribución empregando a inyección (2.21). Por outra banda, para demostrar a converxencia a Delta de Dirac no espazo de distribucións $\mathcal{D}'$, o que temos que ver e que, dada $\varphi \in \mathcal{D}$, entón:

$$\lim_{n\to\infty} \int_{\mathbb{R}} h_n(x)\varphi(x)\,dx = \varphi(0).$$

Agora ben, empregando as propiedades das funcións $\{h_n\}_{n\in\mathbb{N}}$, en particular o feito de que $\displaystyle\int_{\mathbb{R}} h_n(x)\,dx = 1$, terase que a identidade anterior é equivalente a seguinte:

$$\lim_{n\to\infty} \int_{\mathbb{R}} h_n(x)(\varphi(x) - \varphi(0))\,dx = 0.$$

Posto que $\varphi \in \mathcal{D}$, en particular, grazas o teorema do valor medio do cálculo diferencial, $\varphi(x) - \varphi(0) = \varphi'(c_x)\,x$, con $c_x \in (0,x)$ ou $(x,0)$ segundo proceda. Tendo en conta esto último:

$$\int_{\mathbb{R}} |h_n(x)\,(\varphi(x) - \varphi(0))\,|\,dx \leq \|\varphi'\|_{L^\infty(\mathbb{R})} \int_{\mathbb{R}} |x|\, h_n(x)\,dx,$$

onde, neste caso, $\|\varphi'\|_{L^\infty(\mathbb{R})} = \sup_{x\in\mathbb{R}} |\varphi'(x)| < \infty$, xa que φ' é, en particular, unha función continua con soporte nun compacto, sendo entón acotada. Da última desigualdade obtense o resultado, xa que:

$$\left| \int_{\mathbb{R}} h_n(x)\varphi(x)\,dx - \varphi(0) \right| \leq \int_{\mathbb{R}} |x|\, h_n(x)\,dx \to 0,$$

cando n tende a infinito. □

Corolario 2.20. *Sexa* $\varphi_r(x) = \dfrac{1}{\sqrt{4\pi r}} \cdot \exp\left(\dfrac{-x^2}{4r}\right)$*, temos que*

$$\lim_{r\to 0} \varphi_r(x) = \delta_0\ en\ \mathcal{D}'. \tag{2.30}$$

Demostración. Vexamos que a función $\varphi_r(x)$ verifica as condicións do teorema anterior:

1. $\varphi_r(x) \geq 0$, $\forall x \in \mathbb{R}$, $\forall r > 0$, esta condición é evidente e non require de proba.

2. $\displaystyle\int_{\mathbb{R}} \varphi_r(x)\,dx = 1$, para demostralo é suficiente con facer o cambio de variable $t = x/\sqrt{4r}$ e empregar o feito de que $\displaystyle\int_{\mathbb{R}} \exp(-t^2)\,dt = \sqrt{\pi}$.

3. Vexamos por último que $\displaystyle\lim_{r\to 0}\int_{\mathbb{R}} |x|\varphi_r(x)\,dx = 0$. Por un lado, suposta a converxencia da integral impropia,

$$\begin{aligned}\int_{\mathbb{R}} |x|\varphi_r(x)\,dx &= -\int_{(-\infty,0]} x\varphi_r(x)\,dx + \int_{[0,\infty)} x\varphi_r(x)\,dx \\ &= \frac{\sqrt{r}}{\sqrt{\pi}}\left[\lim_{M\to\infty}\exp\left(\frac{-x^2}{4r}\right)\right]_{-M}^{0} - \lim_{M\to\infty}\exp\left(\frac{-x^2}{4r}\right)\Bigg]_{0}^{M}\Bigg] \\ &= \frac{2\sqrt{r}}{\sqrt{\pi}}\lim_{M\to\infty}\left[1-\exp\left(\frac{-M^2}{4r}\right)\right] = \frac{2\sqrt{r}}{\sqrt{\pi}}.\end{aligned}$$

   Finalmente:

$$\lim_{r\to 0}\int_{\mathbb{R}} |x|\varphi_r(x)\,dx = \lim_{r\to 0}\frac{2\sqrt{r}}{\sqrt{\pi}} = 0.$$

□

Observación 2.21. Se analizamos a demostración do teorema 2.19, observamos que podemos prescindir da primeira hipótese, sempre e cando reformulemos a terceira nos seguientes termos:

$$\lim_{n\to\infty}\int_{\mathbb{R}} |x\,h_n(x)|\,dx = 0.$$

Esta reformulación do teorema permítenos ampliar a colección de funcións que converxen a Delta de Dirac.

2.4. Obtención da solución asociada ó modelo *Gaussian Plume*

Non debemos esquecer que os coeficientes de difusión dependen altamente das condicións meteorolóxicas, pero debido á dificultade da súa determinación na practica é moi común facer o seguinte cambio de variable na variable independente,

$$r = \frac{1}{u}\int_0^x K(\xi)\,d\xi\ [\mathrm{m}^2]. \tag{2.31}$$

de tal forma que a nova variable é máis sinxelo de axustar no caso experimental como veremos en seccións posteriores.

Comenzamos pois o proceso de cambio de variable sobre o problema (2.9):
Sabemos, gracias o Teorema Fundamental do Cálculo Integral, que se temos $f : [a, b] \subset \mathbb{R} \to \mathbb{R}$ unha función continua e $u, v : [c, d] \to [a, b]$ dúas funcións derivables, entón,

$$\frac{d}{dx}\int_{u(x)}^{v(x)} f(t)\,dt \;=\; f(v(x)) \cdot v'(x) \;+\; f(u(x)) \cdot u'(x), \tag{2.32}$$

para todo $x \in [c, d]$. Se aplicamos o resultado anterior a (2.31),

$$\frac{\partial r}{\partial x}(x) = \frac{1}{u}[K(x) \cdot 1 \;+\; K(0) \cdot 0]. \tag{2.33}$$

Sendo $C(x, y, z) := c(r, x, y)$, aplicando a regra da cadea:

$$\frac{\partial C}{\partial x} \;=\; \frac{\partial c}{\partial r} \cdot \frac{\partial r(x)}{\partial x} \;=\; \frac{\partial c}{\partial r} \cdot \frac{1}{u}K(x). \tag{2.34}$$

Agora xa temos eliminados os coeficientes K da ecuación (2.9) e chegamos a

$$\frac{\partial c}{\partial r} \;=\; \frac{\partial^2 c}{\partial y^2} \;+\; \frac{\partial^2 c}{\partial z^2}. \tag{2.35}$$

As condicións de fronteira son iguais ás do problema (2.9) solo que substituímos x por r. Isto débese a que considerando o coeficiente de difusión como unha cte (única forma na que se pode dar por válida a solución do modelo *Gaussian Plume*) obtemos que $r = \frac{Kx}{u}$.
Resolvemos agora a EDP (2.35).

Aplicamos en primeira instancia o método de "separación de variables"para y e z en (2.35):

$$c(r, x, y) \;=\; \frac{Q}{u}a(r, y) \cdot b(r, z). \tag{2.36}$$

Utilizamos unha pequena variación do que sería o método na súa forma habitual, onde normalmente consideraríamos $c(r, y, z) = f(r)g(y)h(z)$.
Polo tanto

$$\frac{\partial c}{\partial r}(r, y, z) \;=\; \frac{\partial}{\partial r}\left[\frac{Q}{u}a(r, y) \cdot b(r, z)\right] \;=\; \frac{Q}{u}\left(a(r, y) \cdot \frac{\partial b}{\partial r}(r, z) \;+\; b(r, z) \cdot \frac{\partial a}{\partial r}(r, y)\right), \tag{2.37}$$

$$\frac{\partial^2 c}{\partial y^2}(r, y, z) \;=\; \frac{\partial^2}{\partial y^2}\left[\frac{Q}{u}a(r, y) \cdot b(r, z)\right] \;=\; \frac{Q}{u}b(r, z) \cdot \frac{\partial^2 a}{\partial y^2}(r, y), \tag{2.38}$$

$$\frac{\partial^2 c}{\partial z^2}(r, y, z) \;=\; \frac{\partial^2}{\partial z^2}\left[\frac{Q}{u}a(r, y) \cdot b(r, z)\right] \;=\; \frac{Q}{u}a(r, y) \cdot \frac{\partial^2 b}{\partial z^2}(r, z). \tag{2.39}$$

E así obtemos dous problemas particulares para as funcións a e b

$$\frac{\partial a}{\partial r}(r,y) = \frac{\partial^2 a}{\partial y^2}(r,y),\ 0 \leq r < \infty\,,\ -\infty < y < \infty, \tag{2.40}$$

$$a(0,y) = \delta(y),\ a(\infty,y) = 0\,,\ a(r,\infty) = a(r,-\infty) = 0 \tag{2.41}$$

e

$$\frac{\partial b}{\partial r}(r,z) = \frac{\partial^2 b}{\partial z^2}(r,z),\ 0 \leq r < \infty\,,\ -\infty < z < \infty, \tag{2.42}$$

$$b(0,z) = \delta(z-H),\ b(\infty,z) = 0\,,\ b(r,\infty) = 0\,,\ \frac{\partial b}{\partial z}(r,0) = 0. \tag{2.43}$$

A obtención das condicións de contorno para os respectivos problemas é inmediata. Para a primeira, por exemplo, partimos de que

$$c(0,y,z) = \frac{Q}{u}\delta(y)\delta(z-H) = \frac{Q}{u}\cdot a(0,y)\cdot b(0,z), \tag{2.44}$$

e polo tanto

$$a(0,y) = \delta(y), \tag{2.45}$$

$$b(0,z) = \delta(z-H). \tag{2.46}$$

Para o resto de condicións sucede de forma análoga.
En ambos problemas a variable r pode ser considerada como unha *variable temporal*, polo que cando $r = 0$ respreséntanse as condicións iniciais para cada problema.

2.5. Solución *Gaussian Plume*

Imos utilizar a Transformada de Laplace en dominio real para resolver os problemas (2.40) e (2.42). Empecemos resolvendo para $a(r,y)$.

- Tomamos a transformada de Laplace na EDP (2.40) con respecto a r e denotémola por $\hat{a}(\rho,y) := \mathcal{L}_r(a(r,y)) = \displaystyle\int_0^\infty \mathrm{e}^{-\rho r}\cdot a(r,y)\,dr$, sendo ρ a variable transformada. Agora tendo en conta que

$$\mathcal{L}_r\left(\frac{\partial a}{\partial r}\right) = \rho\mathcal{L}_r(a(r,y)) - a(0,y) = \rho\hat{a} - a(0,y), \tag{2.47}$$

$$\mathcal{L}_r\left(\frac{\partial^2 a}{\partial y^2}\right) = \frac{\partial}{\partial y^2}(\mathcal{L}_r(a(r,y))) = \frac{\partial \hat{a}}{\partial y^2}, \tag{2.48}$$

obtemos

$$\rho\hat{a} - a(0,y) = \frac{\partial \hat{a}}{\partial y^2}. \tag{2.49}$$

Se aplicamos a condición incial $a(0,y) = \delta(y)$ obtemos a EDO

$$\frac{\partial \hat{a}}{\partial y^2} - \rho \hat{a} = -\delta(y). \tag{2.50}$$

Seguidamente aplicamos de novo a transformada de Laplace para (2.50) pero esta vez con respecto a y, e denotando $\hat{\hat{a}}(\rho,y) := \mathcal{L}_r(\hat{a}(\rho,y)) = \displaystyle\int_0^\infty \mathrm{e}^{-\eta y} \cdot \hat{a}(\rho,y)\, dy$ con η nova variable da transformada.

Partindo de novo de que

$$\begin{aligned} \mathcal{L}_r\left(\frac{\partial^2 \hat{a}}{\partial y^2}\right) &= \eta^2 \cdot \mathcal{L}_r(\hat{a}(\rho,y)) - \eta \hat{a}(\rho,0) - \frac{\partial \hat{a}}{\partial y}(\rho,0) \\ &= \eta^2 \hat{\hat{a}}(\rho,\eta) - \eta \hat{a}(\rho,0) - \frac{\partial \hat{a}}{\partial y}(\rho,0), \end{aligned} \tag{2.51}$$

$$\mathcal{L}_r(\rho \hat{a}) = \rho \mathcal{L}_r(\hat{a}) = \rho \hat{\hat{a}}(\rho,\eta), \tag{2.52}$$

$$\mathcal{L}_r(\delta(y)) = 1, \tag{2.53}$$

chegamos a unha nova ecuación

$$\eta^2 \hat{\hat{a}}(\rho,\eta) - \eta \hat{a}(\rho,0) - \frac{\partial \hat{a}}{\partial y}(\rho,0) - \rho \hat{\hat{a}}(\rho,\eta) = -1. \tag{2.54}$$

Observación 2.22. Aínda que estamos restrinxindo y para valores $0 \leq y < \infty$ a simetría permitiría estender a solución para $-\infty < y < \infty$.

Tomando en (2.54) $c_1 = \hat{a}(\rho,0)$ e $c_2 = \partial_y \hat{a}(\rho,0) - 1$, podemos escribir

$$\hat{\hat{a}}(\rho,\eta) = \frac{\eta c_1 + c_2}{\eta^2 - \rho}. \tag{2.55}$$

Agora debemos aplicar o proceso de calcular a inversa da transformada de Laplace

$$\begin{aligned} \hat{a}(\rho,y) &= \mathcal{L}_r^{-1}\left\{\frac{\eta c_1 + c_2}{\eta^2 - \rho}\right\} \\ &= c_1 \mathcal{L}_r^{-1}\left\{\frac{\eta}{\eta^2 - \rho}\right\} - c_2 \mathcal{L}_r^{-1}\left\{\frac{1}{\eta^2 - \rho}\right\} \\ &= c_1 \mathcal{L}_r^{-1}\left\{\frac{\eta}{\eta^2 - (\sqrt{\rho})^2}\right\} - c_2 \mathcal{L}_r^{-1}\left\{\frac{1}{\eta^2 - (\sqrt{\rho})^2}\right\}, \end{aligned} \tag{2.56}$$

Obtendo así [1],

$$\begin{aligned} \hat{a}(\rho,y) &= c_1 \cosh(\sqrt{\rho} y) - \frac{c_2}{\sqrt{\rho}} \operatorname{senh}(\sqrt{\rho} y) \\ &= \frac{c_1}{2}\left[\mathrm{e}^{\sqrt{\rho} y} + \mathrm{e}^{-\sqrt{\rho} y}\right] - \frac{c_2}{\sqrt{\rho}}\left[\mathrm{e}^{\sqrt{\rho} y} - \mathrm{e}^{-\sqrt{\rho} y}\right]. \end{aligned} \tag{2.57}$$

[1] Sendo $F(s) = \mathcal{L}(senh(at)) = \frac{a}{s^2 - a^2}$ e $F(s) = \mathcal{L}(cosh(at)) = \frac{s}{s^2 - a^2}$

En base ás condicións de fronteira, podemos supoñer que $\lim_{y\to\infty} \hat{a}(\rho, y) = 0$, e polo tanto obtemos que $c_1 = \dfrac{c_2}{\sqrt{\rho}}$, o cal reduce $\hat{a}$ en

$$\hat{a}(\rho, y) = \frac{c_2}{\sqrt{\rho}} \mathrm{e}^{-\sqrt{\rho} y}. \tag{2.58}$$

Se non temos en conta polo momento que c_2 depende de ρ realizamos de novo a inversa da transformada de Laplace para (2.58) pero esta vez con respecto a ρ, i.e, $a(r, y) = c_2 \mathcal{L}_\rho^{-1}\left(\dfrac{\mathrm{e}^{-\sqrt{\rho} y}}{\sqrt{\rho}}\right)$,

$$a(r, y) = \frac{c_2}{\sqrt{\pi r}} \mathrm{e}^{\frac{-y^2}{4r}}. \tag{2.59}$$

Empregando a propiedade da función de Dirac que probamos no Corolario 2.20, $\delta(y) = \lim_{r\to 0} \dfrac{1}{\sqrt{4\pi r}} \mathrm{e}^{\frac{-y^2}{4r}}$, sobre a condición de inicial (2.41) obtemos

$$\left[a(0, y) = \delta(y)\right] \Rightarrow \left[\lim_{r\to 0} \frac{c_2}{\sqrt{\pi r}} \mathrm{e}^{\frac{-y^2}{4r}} = \lim_{r\to 0} \frac{1}{\sqrt{4\pi r}} \mathrm{e}^{\frac{-y^2}{4r}}\right] \Rightarrow \left[c_2 = \frac{1}{2}\right],$$

e polo tanto

$$a(r, y) = \frac{1}{\sqrt{4\pi r}} \mathrm{e}^{\frac{-y^2}{4r}}. \tag{2.60}$$

- Realizamos de novo este proceso para o problema (2.42) da función b. Primeiro aplicamos a transformada de Laplace con respecto a r tomando $\hat{b}(\rho, y) := \mathcal{L}_r(b(r, z)) = \displaystyle\int_0^\infty \mathrm{e}^{-\rho r} \cdot b(r, z)\, dr$

$$\frac{\delta^2 \hat{b}}{\delta z^2} - \rho \hat{b} = -\delta(z - H), \tag{2.61}$$

e logo aplicamos con respecto a z con $\hat{\hat{b}}(\rho, \xi) := \mathcal{L}_z(b(\rho, z)) = \displaystyle\int_0^\infty \mathrm{e}^{-\rho r} \cdot \hat{b}(\rho, z)\, dr$,

$$\xi^2 \hat{\hat{b}} - \xi \hat{b}(\rho, 0) - \frac{\partial \hat{b}}{\partial z}(\rho, 0) - \rho \hat{\hat{b}}(\rho, \xi) = -\mathrm{e}^{-\xi H}. \tag{2.62}$$

Aplicando a condición fronteira de tipo Neumann $\dfrac{\partial b}{\partial z} b(\rho, 0) = 0$ temos que

$$\hat{\hat{b}}(\rho, \xi) = \frac{\xi \hat{b}(\rho, 0) - \mathrm{e}^{-\xi H}}{\xi^2 - \rho}. \tag{2.63}$$

Agora, aplicando a inversa da transformada de Laplace en ξ obtemos

$$\begin{aligned} \hat{b}(\rho, z) &= \hat{b}(\rho, 0) \cosh(\sqrt{\rho} z) - \tfrac{1}{\sqrt{\rho}} \operatorname{senh}(\sqrt{\rho}(z - H)) \\ &= \hat{b}(\rho, 0) \frac{\mathrm{e}^{\sqrt{\rho} z} + \mathrm{e}^{-\sqrt{\rho} z}}{2} - \frac{1}{\sqrt{\rho}} \frac{\mathrm{e}^{\sqrt{\rho}(z-H)} + \mathrm{e}^{-\sqrt{\rho}(z-H)}}{2}. \end{aligned} \tag{2.64}$$

Impoñendo de novo que $\lim_{z\to\infty} \hat{b}(\rho, z) = 0$,

$$\begin{aligned} \lim_{z\to\infty} \hat{b}(\rho,0)\frac{\mathrm{e}^{\sqrt{\rho}z} + \mathrm{e}^{-\sqrt{\rho}z}}{2} - \frac{1}{\sqrt{\rho}}\frac{\mathrm{e}^{\sqrt{\rho}(z-H)} + \mathrm{e}^{-\sqrt{\rho}(z-H)}}{2} = \\ = \hat{b}(\rho,0)\frac{\mathrm{e}^{\sqrt{\rho}z}}{2} - \frac{1}{2\sqrt{\rho}}\mathrm{e}^{\sqrt{\rho}z} + \mathrm{e}^{-\sqrt{\rho}H} = 0, \end{aligned} \tag{2.65}$$

obtemos $\hat{b}(\rho, 0) = \frac{1}{\sqrt{\rho}}\mathrm{e}^{-\sqrt{\rho}H}$ e substituíndo en (2.64)

$$\hat{b}(\rho, z) = \frac{1}{\sqrt{\rho}}\mathrm{e}^{-\sqrt{\rho}H}\frac{\mathrm{e}^{\sqrt{\rho}z} + \mathrm{e}^{-\sqrt{\rho}z}}{2} - \frac{1}{\sqrt{\rho}}\frac{\mathrm{e}^{\sqrt{\rho}(z-H)} + \mathrm{e}^{-\sqrt{\rho}(z-H)}}{2}, \tag{2.66}$$

que pode simplificarse como

$$\hat{b}(\rho, z) = \frac{1}{2\sqrt{\rho}}\left(\mathrm{e}^{-\sqrt{\rho}(z-H)} + \mathrm{e}^{-\sqrt{\rho}(z+H)}\right). \tag{2.67}$$

Finalmente, aplicando a inversa da transformada de Laplace sobre ρ en (2.67)

$$b(r, z) = \frac{1}{\sqrt{4\pi r}}\left(\mathrm{e}^{-\frac{(z-H)^2}{4r}} + \mathrm{e}^{-\frac{(z+H)^2}{4r}}\right). \tag{2.68}$$

Agora que xa temos as expresións para a(r,y) e b(r,z)[2], (2.60) e (2.68) respectivamente, xa podemos substituílas en (2.36), expresión da concentración do contaminante

$$c(r, y, z) = \frac{Q}{4\pi u r}\,\mathrm{e}^{-\frac{y^2}{4r}}\,(\mathrm{e}^{-\frac{(z-H)^2}{4r}} + \mathrm{e}^{-\frac{(z+H)^2}{4r}}). \tag{2.69}$$

Esta solución é a coñecida como *Gaussian Plume*.

2.6. Efectos da variable de difusión

No caso de tratar cun modelo máis xeral co *Gaussian Plume*, debemos sempre ter en conta a especificación do coeficiente de difusión $K(x)$. Dado que no modelo requirimos que K sexa constante, agora debemos permitir que K varíe a favor do vento (con respecto ó eixo x) e que o seu valor se obteña de forma precisa tendo en conta as observacións experimentais.

É habitual no estudio desta materia empezar substituíndo a variable r en (2.69) por

$$\sigma^2(x) = \frac{2}{u}\int_0^x K(\xi)\,d\xi = 2r, \tag{2.70}$$

onde $\sigma(x)$ se coñece como unha *desviación estándar* de la concentración (Gaussiana).

[2]Podemos atopar outras expresións para a e b, obtidas mediante métodos distintos, nas bibliografías [10] [7].

Estes coeficientes son máis sinxelos de determinar de forma experimental que os coeficientes de difusión K e adquiren numerosas versións de representación. A máis común ven dada por unha lei moi sinxela, $\sigma^2 = ax^b$.
As posibles estimacións para a e b están baseadas en diversos procesos experimentais que teñen en conta a variedade de condicións atmosféricas.

Unha posible **interpretación xeométrica** para σ: unha vez nos afastamos da fonte, o penacho (*plume*) de contaminación se ensancha e polo tanto o valor de σ vese incrementado. Este ancheamento é debido á doble dirección do vento, con respecto ó eixo z e ó eixo y, como podemos observar na figura 2.1.

De acordo coa definición[3] de σ, (2.70), o coeficiente de difusión pode escribirse como $K(x) = \frac{1}{2}u\frac{\partial}{\partial x}\sigma^2$. No caso para K cte, obtemos directamente de (2.70) que $\sigma^2 = \frac{2Kx}{u}$.

Observación 2.23. É moi común na práctica denotar $K(x) = \dfrac{u\sigma^2}{2x}$ como o coeficiente de difusión, e será esta a notación que se empregará de aquí en adiante.

Partindo de (2.70) de novo, supoñendo K cte , se integramos, obtemos $r = \dfrac{Kx}{u}$ e substituindo en (2.69) acadamos unha nova expresión para a concentración de contaminante

$$C(x,y,z) = \frac{Q}{2\pi K}\,\mathrm{e}^{-u\frac{y^2}{4Kx}}\left(\mathrm{e}^{-u\frac{(z-H)^2}{4Kx}} + \mathrm{e}^{-u\frac{(z+H)^2}{4Kx}}\right). \tag{2.71}$$

Facendo a simplificación da expresión anterior para $z = 0$ estariamos a calcular a concentración ao nivel do chan,

$$C(x,y,0) = \frac{Q}{2\pi K}\,\mathrm{e}^{-u\frac{(y^2+H^2)}{4Kx}}. \tag{2.72}$$

Esta simplificación permite facer unha validación da solución obtida no modelo *Gaussian Plume*, xa que podemos comprobar se se verifica que para $z = 0$ o punto onde hai un maior nivel de concentración é en $H = 0$, como poderiamos esperar de forma intuitiva.

Cando o punto fonte localizado en $(0,0,H)$ está elevado ($H > 0$) somos capaces de calcular o punto onde a súa concentración é máxima e o valor que acada.

Comenzamos derivando a ecuación (2.72) con respecto a x

$$\begin{aligned}\frac{\partial C}{\partial x}(x,y,0) &= \frac{Q\cdot 2\pi K}{(2\pi K)^2}\cdot \mathrm{e}^{-u\frac{(y^2+H^2)}{4Kx}} + \frac{Q}{2\pi K}\cdot -u\frac{(y^2+H^2)4K}{(4Kx)^2}\cdot \mathrm{e}^{-u\frac{(y^2+H^2)}{4Kx}} \\ &= \frac{Q}{2\pi K}\mathrm{e}^{-u\frac{(y^2+H^2)}{4Kx}}\cdot\left(\frac{1}{x^2} + \frac{-u(y^2+H^2)}{4Kx^3}\right).\end{aligned} \tag{2.73}$$

[3]Existen outras formas de denotar sigma, [14].

Procedemos agora a buscar un punto de máximo igualando a ecuación anterior (2.73) a cero. Despexando x obtemos que $x = u\frac{H^2}{4K}$ sendo este o valor onde a concentración é máxima, $x_{\text{máx}}$. Substituíndo este valor de $x_{\text{máx}}$ na ecuación (2.72) (tendo en conta $y = 0$ e $z = 0$) obtemos que a concentración máxima en $(0, 0, H)$ alcanza un valor de $C_{\text{máx}} = \dfrac{2Q}{(\pi u H^2 \text{e})}$

Cabe destacar que se facemos tender a velocidade de dispersión a cero

$$\lim_{u \to 0^+} C(x, y, z) = \frac{Q}{2\pi K x}, \tag{2.74}$$

o cal pode chegar a contradicir a idea analítica de que a solución do modelo *Gaussian Plume* "se rompe"cando $u = 0$. Esta confusión normalmente está asociada á elección de (2.69), é dicir, tomando a solución en términos da variable r, na cal aparece unha singularidade cando $u \longrightarrow 0$ se esquecemos a dependencia de r sobre a variable velocidade, $r = \frac{1}{u}\int_0^x K(\xi)\, d\xi$.

Agora ben, é importante lembrar que a solución do modelo *Gaussian Plume* só ten sentido físico cando a velocidade do vento é distinta de cero debido á suposición número 5 (2.5), onde desprezábamos os efectos difusivos no eixo x.

Cando queremos considerar seriamente a condición $u = 0$[4], a relación da concentración cos efectos de difusión ven dada baixo a **Ecuación estable de difusión** (*Ecuación de Poisson*)

$$\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2} + \frac{\partial^2 C}{\partial z^2} = -\frac{Q}{K}\delta(x)\delta(y)\delta(z - H), \tag{2.75}$$

con solución

$$C(x, y, z) = \frac{Q}{4\pi K}\left(\frac{1}{\sqrt{x^2 + y^2 + (z - H)^2}} + \frac{1}{\sqrt{x^2 + y^2 + (z - H)^2}}\right), \tag{2.76}$$

para o semi-espazo $x > 0$.

2.7. Variantes da solución do modelo *Gaussian Plume*

Presentamos nesta sección algunhas das variantes que podemos obter a partir do Modelo *Gaussian Plume* se consideramos distintas suposicións, tanto para o medio como para as condicións iniciais e de fronteira.

2.7.1. *Anisotropic Eddy Diffusivities*

Neste caso estamos considerando que os coeficientes de difusión K_y e K_z son distintos, $K_y \neq K_z$, e definidos de forma que

$$r_i = \frac{1}{u}\int_0^x K_i(\xi)\, d\xi, \;\; i \in \{y, z\}.$$

[4]Coñecido como modelo *low-wind*, [3].

Usando o procedemento xa descrito anteriormente obtemos,

$$C(x,y,z) = \frac{Q}{4\pi u\sqrt{r_y r_z}}\, \mathrm{e}^{(-\frac{y^2}{4r_y})} \left[\mathrm{e}^{\left(-\frac{(z-H)^2}{4r_z}\right)} + \mathrm{e}^{\left(-\frac{(z+H)^2}{4r_z}\right)} \right]. \tag{2.77}$$

2.7.2. Absorción perfecta do chan

Aquí imos remplazar a condición impermeabilidade (2.6) por unha condición de absorción por parte do chan, $c(r,y,0) = 0$. Aplicando algunhas modificacións no proceso para $b(r,y)$ obtemos,

$$c(r,y,z) = \frac{Q}{4\pi ur}\, \mathrm{e}^{\left(-\frac{y^2}{4r}\right)} \left[\mathrm{e}^{\left(-\frac{(z-H)^2}{4r}\right)} + \mathrm{e}^{\left(-\frac{(z+H)^2}{4r}\right)} \right]. \tag{2.78}$$

2.7.3. Capa de inversión

No límite da capa atmosférica a temperatura do aire decrece a medida que aumentamos a altitude; sen embargo, existen capas onde se produce o efecto contrario, chamadas capas de inversión. Nelas a temperatura do aire vese incrementada coa altitude e polo tanto a concentración é estable e resistente con respecto á mestura vertical. Cando se dan este tipo de capas prodúcese unha especie de néboa tóxica sobre a poboación.

(a) Los Ángeles, 26 de xullo de 1943 [18].

(b) México, 2 de xuño de 2018.(Fotografía de Armando Monroy [4]).

Figura 2.3: Dous exemplos de cidades que sofren o efecto das capas de inversión.

Se substituímos a condición tipo Dirichlet $c(r, y, \infty) = 0$ por unha nova condición de Neumann $K\frac{\partial}{\partial z}c(r, y, D) = 0$, sendo D a altura á que se produce a capa de inversión. Facendo unha modificación nas condicións de contorno para o problema $b(r, y)$ obtemos a nova solución [5]:

$$c(r, y, z) = \frac{Q}{uD\sqrt{\pi r}} \cdot \mathrm{e}^{-\frac{y^2}{4r}} \left[\frac{1}{2} + \sum_{n=1}^{\infty} \cos\left(\frac{n\pi z}{D}\right) \cos\left(\frac{n\pi H}{D}\right) \mathrm{e}^{-r\left(\frac{n\pi}{D}\right)^2}\right]. \tag{2.79}$$

2.7.4. Liñas fonte

Para os estudios sobre a emisión de contaminación dos coches en estradas moi concorridas os modelos de dispersión poden ser aproximados por liñas fonte.

As carreteiras que son longas, rectilíneas e que teñen dirección perpendicular á dirección do vento poden ser representadas por unha liña fonte de lonxitude infinita sobre o eixo y, o que ven a corresponderse coa condición de contorno $c(0, y, z) = \frac{Q_L}{u}\delta(z)$. A constante Q_L [$\mathrm{kg\,m^{-1}\,s^{-1}}$] representa o grado de emisión por unidade de lonxitude da carreteira. (Non confundir coa constante Q).
Coa nova condición de contorno obtemos a solución ,

$$c(r, y, z) = \frac{Q_L}{2\pi ur} \cdot \mathrm{e}^{-\frac{z^2}{4r}} \int_{-\infty}^{\infty} \mathrm{e}^{\left(-\frac{y^2}{4r}\right)}\, dy = \frac{Q_L}{\sqrt{\pi u}} \cdot \mathrm{e}^{-\frac{z^2}{4r}}. \tag{2.80}$$

Un caso máis realista sería considerar unha carreteira de extensión finita L, [6], tomando a mesma condición de contorno para $|y| \leq L/2$ e $c(0, y, z) = 0$ noutro caso:

$$c(r, y, z) = \frac{Q_L}{2u\sqrt{\pi r}} \cdot \mathrm{e}^{-\frac{z^2}{4r}} \left[erf\left(\frac{y + L/2}{2\sqrt{r}}\right) - erf\left(\frac{y - L/2}{2\sqrt{r}}\right)\right], \tag{2.81}$$

onde $erf(x) = 2\pi^{-1/2} \int_0^x \mathrm{e}^{-\xi^2}\, d\xi$ é a función de erro.

[5] Para máis información do proceso de construción da solución asociada a unha capa de inversión consultar a bibliografía [1].

Capítulo 3

Resolución numérica

Neste capítulo presentaremos a resolución numérica do problema de optimización, co obxectivo de determinar o mellor dos camiños (en función da concentración do contaminate e a distancia percorrida) para a neutralización dun incidente químico. Utilizaremos a ferramenta MATLAB, cuxo código asociado pode atoparse no apéndice.

Primeiramente debemos expoñer o problema debidamente. Para iso, seguiremos o seguinte esquema: na primeira sección describiremos o problema, detallando os distintos elementos que o compoñen (a variable de control, o funcional de coste e a ecuación de estado). Na segunda sección veremos como resolver o problema e, na sección terceira, veremos unha serie de resultados numéricos.

3.1. Descrición do problema

Representamos a zona afectada por contaminación sobre a que queremos traballar como o conxunto $\{(x, y, Href), (x, y) \in \Omega\}$ sendo Ω un dominio $\Omega \subset \mathbb{R}^2$ e Href a altura mínima esixida para os bombeiros, tanto mulleres como homes, en España.

O noso obxectivo é atopar un *camiño seguro* que minimice o grao de contaminación ao que se ven sometidos os bombeiros (ou o equipo designado) no proceso de neutralización dun incidente químico. Para parametrizar dito camiño, podemos supoñer que se trata dunha curva parametrizada regular a trozos, de forma que cada un dos trozos correspóndese cun segmento de extremos a determinar. Desta forma, se denotamos por $\{(x_i, y_i)\}_{i \in I} \subset \Omega$ os puntos extremos dos devanditos segmentos (I será o conxunto de índices que detallaremos máis adiante) e supoñemos que o dominio ven dado por $\Omega = [0, L_1] \times \left[\frac{-L_2}{2}, \frac{L_2}{2}\right]$, o problema

pódese escribir nos seguintes termos:

$$\begin{aligned} \min_{(x,y)\in\Omega} \quad & f(\vec{x},\vec{y}) \\ \text{suxeito a} \quad & x_i \in [0, L_1], \quad \forall\, i \in I, \\ & y_i \in \left[\frac{-L_2}{2}, \frac{L_2}{2}\right], \quad \forall\, i \in I, \\ & (x_i, y_i) \neq (x_j, y_j), \text{ se } i \neq j, \end{aligned} \tag{3.1}$$

onde $f : \mathbb{R}^{\#I+2} \times \mathbb{R}^{\#I+2} \longrightarrow \mathbb{R}$ é o funcional de coste (que terá en conta a distancia percorrida e a dose de contaminante á que están expostos os bombeiros ao longo do camiño percorrido), que podemos supor continuamente diferenciable. Tanto o punto de saída (x_0, y_0), como o punto de chegada $(x_{\#I+1}, y_{\#I+1})$ (punto no que se deberán situar os bombeiros para neutralizar o axente químico), supoñeranse coñecidos, así como o punto no que se produce o incidente, as características da sustancia química, e as variables meteorolóxicas que inflúen na sua dispersión.

3.1.1. Variables de control

Supoñamos entón que $I = \{1, \ldots, n\}$, o noso problema de minimización terá como obxectivo a obtención das que consideraremos como *variables de control*, $(x_1, y_1), \ldots, (x_n, y_n)$, que determinan as posicións que deben ir seguindo os profesionais sobre o dominio.

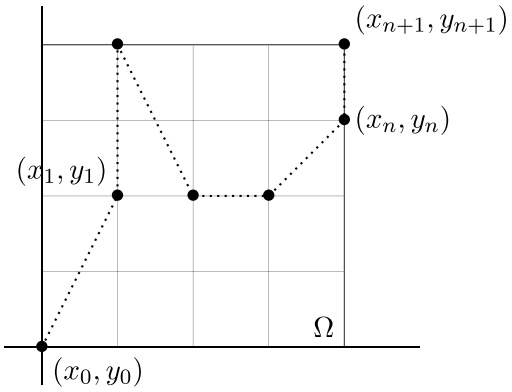


Figura 3.1: Representación gráfica do problema de minimización

Neste traballo imos parametrizar a curva $\sigma_{\vec{x},\vec{y}}$, que representará o camiño a seguir, en forma de liña poligonal, onde as variables de control van ser os punto de inicio e final de cada segmento de recta que compoñen a curva. Tomaremos $\vec{x} = (x_0, x_1, \ldots, x_n, x_{n+1})$ e $\vec{y} = (y_0, y_1, \ldots, y_n, y_{n+1})$ e definimos $\sigma_{\vec{x},\vec{y}}$ tal que:

$$\sigma_{\vec{x},\,\vec{y}}\,(t) = \begin{cases} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + (t-0)\left(\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right) & 0 \le t \le 1 \\ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + (t-1)\left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}\right) & 1 \le t \le 2 \\ \vdots & \vdots \\ \begin{pmatrix} x_n \\ y_n \end{pmatrix} + (t-n)\left(\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} - \begin{pmatrix} x_n \\ y_n \end{pmatrix}\right) & n \le t \le n+1 \end{cases} \tag{3.2}$$

como vemos representada na figura 3.1

3.1.2. Funcional de coste

Buscamos agora esa función f diferenciable e continua que representa o funcional de coste nos puntos $(x, y, Href)$, con $(x, y) \in \Omega$. Para construír este funcional de coste teremos en conta as seguintes achegas:

1. Achega debida á concentración de contaminante sobre a curva, $g(\vec{x}, \vec{y})$.

2. Achega debida á distancia total percorrida.

Con respecto a primeira das aportacións, é necesario calcular a concentración sobre a curva $\sigma_{\vec{x},\,\vec{y}}\,(t)$:

$$g(\vec{x}, \vec{y}) = \int_{\sigma_{\vec{x},\,\vec{y}}} C\, ds = \sum_{k=0}^{n} \int_{k}^{k+1} C(\sigma_{\vec{x},\,\vec{y}}\,(t)) \|\sigma'_{\vec{x},\,\vec{y}}\,(t)\|\, dt, \tag{3.3}$$

desenrolando esta expresión,

$$\begin{aligned} g(\vec{x}, \vec{y}) &= \sum_{k=0}^{n} \int_{k}^{k+1} C(\sigma_{\vec{x},\vec{y}}(t)) \parallel \sigma_{\vec{x},\vec{y}}(t) \parallel\, dt \\ &= \sum_{k=0}^{n} \int_{k}^{k+1} C_k(t) \cdot \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}\, dt \qquad (3.4) \\ &= \sum_{k=0}^{n} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \int_{k}^{k+1} C_k(t)\, dt, \end{aligned}$$

onde

$$C_k(t) = C\left(x_k + (t-k)(x_{k+1} - x_k), y_k + (t-k)(y_{k+1} - y_k)\right), \tag{3.5}$$

é a concentración no segmento k-ésimo. Agora ben, xa temos calculada a solución do modelo Gaussian Plume, (2.69), que está en función de r, y e z, e polo tanto temos que desfacer o cambio (2.31) en r, de tal forma que si consideramos K cte:

$$r = \frac{1}{u}\int_0^x K(\xi)\,d\xi = \frac{K}{u}\,x \;\Rightarrow\; x = \frac{ru}{K},$$

e polo tanto a expresión da concentración ven dada como:

$$C(x,y,z) = \frac{Q}{4\pi K x}\mathrm{e}^{-\frac{uy^2}{4Kx}}\left[\mathrm{e}^{-u\frac{(z-H)^2}{4Kx}} + \mathrm{e}^{-u\frac{(z+H)^2}{4Kx}}\right]. \tag{3.6}$$

Consideramos agora que $z = Href$, xa que intentaremos determinar o camiño seguro en base a esa altura, polo que os valores da concentración pasan só a depender de x e y, $C(x,y)$:

$$C(x,y) = \frac{Q}{4\pi K x}\mathrm{e}^{-\frac{uy^2}{4Kx}}\left[\mathrm{e}^{-u\frac{(Href-H)^2}{4Kx}} + \mathrm{e}^{-u\frac{(Href+H)^2}{4Kx}}\right]. \tag{3.7}$$

Durante o proceso de resolución do problema (3.1) tamén imos precisar as parciais de concentración C con respecto a x e y:

$$\begin{aligned}
\frac{\partial C}{\partial x} &= \frac{Q}{4\,K\,x\,\pi}\mathrm{e}^{-\frac{u\,y^2}{4\,K\,x}}\left(\frac{u\,\mathrm{e}^{-\frac{u\,(H-\mathrm{Href})^2}{4\,K\,x}}\,(H-\mathrm{Href})^2}{4\,K\,x^2} + \frac{u\,\mathrm{e}^{-\frac{u\,(H+\mathrm{Href})^2}{4\,K\,x}}\,(H+\mathrm{Href})^2}{4\,K\,x^2}\right)\\
&- \frac{Q}{4\,K\,x^2\,\pi}\mathrm{e}^{-\frac{u\,y^2}{4\,K\,x}}\left(\mathrm{e}^{-\frac{u\,(H+\mathrm{Href})^2}{4\,K\,x}} + \mathrm{e}^{-\frac{u\,(H-\mathrm{Href})^2}{4\,K\,x}}\right)\\
&+ \frac{Q}{8\,K^2\,x^2\,\pi}\,u\,y^2\,\mathrm{e}^{-\frac{u\,y^2}{4\,K\,x}}\left(\mathrm{e}^{-\frac{u\,(H+\mathrm{Href})^2}{4\,K\,x}} + \mathrm{e}^{-\frac{u\,(H-\mathrm{Href})^2}{4\,K\,x}}\right),\\
\frac{\partial C}{\partial y} &= -\frac{Q}{8\,K^2\,x^2\,\pi}\,u\,y\,\mathrm{e}^{-\frac{u\,y^2}{4\,K\,x}}\left(\mathrm{e}^{-\frac{u\,(H+\mathrm{Href})^2}{4\,K\,x}} + \mathrm{e}^{-\frac{u\,(H-\mathrm{Href})^2}{4\,K\,x}}\right).
\end{aligned} \tag{3.8}$$

Para calcular a segunda das achegas, simplemente teremos en conta que a suma

$$\sum_{k=0}^{n}\sqrt{(x_{k+1}\,-\,x_k)^2\,+\,(y_{k+1}\,-\,y_k)^2} \tag{3.9}$$

representa a distancia total recorrida. Polo tanto, podemos definir a función de coste, da seguinte forma:

$$\begin{aligned}
f : \mathbb{R}^{n+2}\times\mathbb{R}^{n+2} &\longrightarrow \mathbb{R},\\
\vec{w} = (\vec{x},\vec{y}) &\longrightarrow f(\vec{w}),
\end{aligned} \tag{3.10}$$

onde:

$$f(\vec{w}) = \lambda_1\cdot g(\vec{w}) \,+\, \lambda_2\cdot\sum_{k=0}^{n}\sqrt{(x_{k+1}-x_k)^2+(y_{k+1}-y_k)^2}, \tag{3.11}$$

con λ_1, $\lambda_2 \in \mathbb{R}$ dous números reais tales que $\lambda_1 + \lambda_2 = 1$. Ditos parámetros posibilitarán un estudio máis pormenorizado da influencia dos termos anteriores nos resultados obtidos.

Debemos ter en conta que o cálculo da integral do termo de concentración non é sinxela, polo que o emprego dunha fórmula de cuadratura, como, por exemplo a *Regra do Trapecio Simple* [1], destaca como unha opción viable:

$$g(\vec{x},\vec{y}) = \sum_{k=0}^{n} \sqrt{(x_{k+1}-x_k)^2 + (y_{k+1}-y_k)^2}\, \frac{1}{2}\left[C(x_k,y_k) + C(x_{k+1},y_{k+1})\right]. \tag{3.12}$$

3.2. Aproximación da solución

Para proceder á resolución do problema (3.1) faremos uso de técnicas numéricas de optimización, máis en concreto imos utilizar o Método do Gradiente Conxugado con Proxección Ortogonal, que podemos consultar na bibliografía [5].
Para que un problema de minimización da forma

$$\begin{array}{ll} \text{mín} & f(x) \\ \text{suxeito a} & x \in X \end{array} \tag{3.13}$$

cumpra as condicións de converxencia do método tense que verificar que,

- X é un subconxunto de $\mathbb{R}$ convexo non vacío.
- A función f é continua e diferenciable sobre o conxunto X.

O proceso iterativo deste método para atopar unha solución x^*, para funcións en $\mathbb{R}$ (non cuadráticas), que minimice o valor da función nun dominio convexo e compacto, ven dado por:

$$x^{k+1} = x^k + \alpha^k \left(\overline{x}^k - x^k\right), \tag{3.14}$$

con

$$\overline{x}^k = P_\Omega(x^k - s^k \nabla f(x^k)) \quad \text{(proxección ortogonal sobre o dominio } \Omega), \tag{3.15}$$

para $s^k \in \mathbb{R}^+$, $\alpha^k \in (0,1]$. Existen varias formas de selección tanto de α^k como do paso s^k. Neste traballo imos tomar un $\alpha^k = 1$, $\forall\, k$ e $s^k = s\,\text{cte}$, $\forall\, k = 0, 1, \ldots$, polo que (3.14) queda:

$$\overline{x}^k = x^{k+1} = P_\Omega(x^k - s\nabla f(x^k)). \tag{3.16}$$

[1] A regra do Trapecio Simple é un método para aproximar o valor dunha integral definida $\int_a^b f$ mediante a integral sobre a función lineal que define a recta que pasa por $(a, f(a))$ e $(b, f(b))$, $\int_a^b \left[f(a) + \frac{f(b)-f(a)}{b-a}(x-a)\right] dx$. De tal forma que $\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$.

O proceso iterativo termina cando atopamos o mínimo da función [2]. O test de parada que introducimos no algoritmo para probar que se verifica que $x^* \in X$ é mínimo pode definirse de diversas formas. Na seguinte subsección especificaremos estes tests.

3.2.1. Algoritmo para a resolución do problema

Para o noso problema de minimización temos que axustar o método para unha función en varias variables, $f : \mathbb{R}^{n+2} \times \mathbb{R}^{n+2} \longrightarrow \mathbb{R}$.

O punto inicial (x_0, y_0) e punto final (x_{n+1}, y_{n+1}) se consideran fixos, e denotamos $\vec{x'} = (x_1, \ldots, x_n)$ e $\vec{y'} = (y_1, \ldots y_n)$ de tal forma que $\vec{w'} = (\vec{x'}, \vec{y'})$. Tomando $s = d \in \mathbb{R}^+$ como paso de descenso, o método iterativo ven dado por:

$$(\vec{x'}^{\,n+1}, \vec{y'}^{\,n+1}) = P_\Omega \left((\vec{x'}^{\,n}, \vec{y'}^{\,n}) - d\nabla f(\vec{x}^{\,n}, \vec{y}^{\,n}) \right). \tag{3.17}$$

Imos construír 3 tests de parada simples para o método iterativo. Tomando unha tolerancia, tol, como un valor real suficientemente pequeno, temos:

- Gradiente nulo,

$$\| \nabla f(\vec{x}, \vec{y}) \|_2 < tol. \tag{3.18}$$

- Diferencia de iterantes consecutivos,

$$\| (\vec{x'}^{\,n+1}, \vec{y'}^{\,n+1}) - (\vec{x'}^{\,n}, \vec{y'}^{\,n}) \|_2 < tol. \tag{3.19}$$

- Diferencia no coste asociado a iterantes consecutivos,

$$|f(\vec{x^{n+1}}, \vec{y^{n+1}}) - f(\vec{x^n}, \vec{y^n})| < tol. \tag{3.20}$$

Para o proceso do cálculo iterativo imos precisar o valor da compoñente k-ésima do vector gradiente ∇f que ven dada pola seguinte expresión:

$$\begin{aligned}
(\nabla f)_k &= \frac{\lambda_1}{2}\left[\frac{\partial C}{\partial x_k}(x_k, y_k)\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} + \frac{\partial C}{\partial x_k}(x_k, y_k)\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \right. \\
&+ \left. \frac{C(x_{k-1}, y_{k-1}) + C(x_k, y_k)}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} \cdot (x_k - x_{k-1}) - \frac{C(x_k, y_k) + C(x_{k+1}, y_{k+1})}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \cdot (x_{k+1} - x_k) \right] \\
&+ \lambda_2 \left[\frac{1}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} \cdot (x_k - x_{k-1}) - \frac{1}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \cdot (x_{k+1} - x_k) \right],
\end{aligned} \tag{3.21}$$

[2] Consideramos que x^* é mínimo global de f se $f(x^*) \leq f(x), \ \forall x \in X$. Diremos que x^* é mínimo local se existe $\varepsilon > 0$ tal que $f(x^*) \leq f(x)$ para $\|x^* - x\| < \varepsilon$.

para valores de $k = 1 \dots n$. Para valores de $k = 1, \dots, n$, temos a seguinte expresión nas correspondentes compoñentes $(\nabla f)_{k+n}$:

$$\begin{aligned}(\nabla f)_{k+n} &= \frac{\lambda_1}{2}\left[\frac{\partial C}{\partial y_k}(x_k, y_k)\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} + \frac{\partial C}{\partial y_k}(x_k, y_k)\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}\right. \\ &+ \left.\frac{C(x_{k-1}, y_{k-1}) + C(x_k, y_k)}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} \cdot (y_k - y_{k-1}) - \frac{C(x_k, y_k) + C(x_{k+1}, y_{k+1})}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \cdot (y_{k+1} - y_k)\right] \\ &+ \lambda_2\left[\frac{1}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} \cdot (y_k - y_{k-1}) - \frac{1}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \cdot (y_{k+1} - y_k)\right].\end{aligned} \tag{3.22}$$

Para o proceso de cálculo da proxección ortogonal $P_\Omega(\,\cdot\,)$, no método iterativo (3.17), imos proceder diferenciando varios casos, como podemos ver na figura 3.2, dependendo de onde se atope o punto (x, y) que debemos proxectar sobre o dominio $\Omega = [0, L_1] \times \left[\frac{-L_2}{2}, \frac{L_2}{2}\right]$, sendo L_1 e L_2 as medidas horizontal e vertical do dominio, respectivamente.

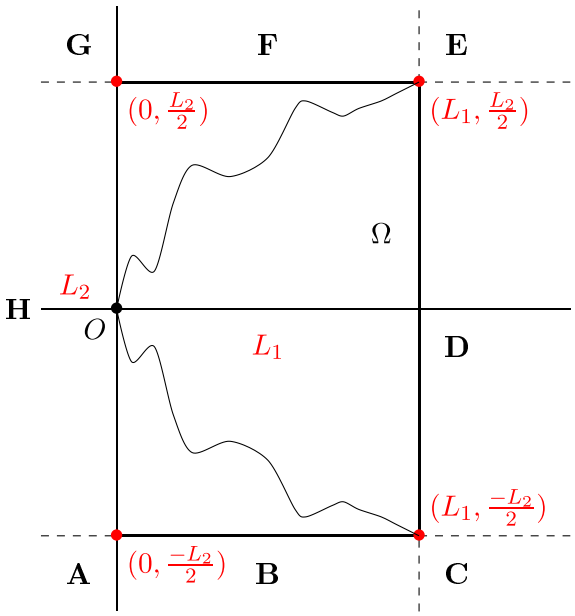


Figura 3.2: Representación gráfica da distribución do contaminante sobre o dominio Ω, considerando O como punto fonte e L_2 e L_1 as dimensións do dominio.

Sendo $P_\Omega : \mathbb{R}^2 \longrightarrow \Omega$ a proxección ortogonal sobre o dominio $\Omega \subset \mathbb{R}^2$ temos:

- **Caso A**

$$\left.\begin{array}{l} -\infty < x \leq 0 \\ -\infty < y \leq \frac{-L_2}{2} \end{array}\right\} \Rightarrow P_\Omega(x, y) = \left(0, \frac{-L_2}{2}\right) \tag{3.23}$$

- **Caso B**

$$\left.\begin{array}{l} 0 < x \leq L_1 \\ -\infty < y \leq \frac{-L_2}{2} \end{array}\right\} \Rightarrow P_\Omega(x,y) = \left(x, \frac{-L_2}{2}\right) \tag{3.24}$$

- **Caso C**

$$\left.\begin{array}{l} L_1 < x \leq \infty \\ -\infty < y \leq \frac{-L_2}{2} \end{array}\right\} \Rightarrow P_\Omega(x,y) = \left(L_1, \frac{L_2}{2}\right) \tag{3.25}$$

- **Caso D**

$$\left.\begin{array}{l} L_1 < x < \infty \\ \frac{-L_2}{2} < y \leq \frac{L_2}{2} \end{array}\right\} \Rightarrow P_\Omega(x,y) = (L_1, y) \tag{3.26}$$

- **Caso E**

$$\left.\begin{array}{l} L_1 < x < \infty \\ \frac{L_2}{2} < y < \infty \end{array}\right\} \Rightarrow P_\Omega(x,y) = \left(L_1, \frac{L_2}{2}\right) \tag{3.27}$$

- **Caso F**

$$\left.\begin{array}{l} 0 < x \leq L_1 \\ \frac{L_2}{2} < y < \infty \end{array}\right\} \Rightarrow P_\Omega(x,y) = \left(x, \frac{L_2}{2}\right) \tag{3.28}$$

- **Caso G**

$$\left.\begin{array}{l} -\infty < x \leq 0 \\ \frac{L_2}{2} < y < \infty \end{array}\right\} \Rightarrow P_\Omega(x,y) = \left(0, \frac{L_2}{2}\right) \tag{3.29}$$

- **Caso H**

$$\left.\begin{array}{l} -\infty < x \leq 0 \\ \frac{-L_2}{2} < y \leq \frac{L_2}{2} \end{array}\right\} \Rightarrow P_\Omega(x,y) = (0, y) \tag{3.30}$$

Para o caso trivial no que o punto $(x, y) \in \Omega$ temos que $P_{R_{|\Omega}} = Id_\Omega$, sendo Id_Ω a función identidade en Ω, i.e, $Id_\Omega : \Omega \longrightarrow \Omega$ con $Id_\Omega(x, y) = (x, y)$.

3.3. Resultados numéricos

Unha vez definido o problema de minimización e descrito todo o proceso que debemos seguir para chegar a unha solución óptima, procedemos a súa resolución numérica utilizando a ferramenta MATLAB. O programa principal e as funcións auxiliares que imos utilizar nesta sección atópanse no Anexo.

Implementamos tanto as funcións de concentración e de coste, *conc.m* e *funcoste.m*, como os seus respectivos gradientes, *gradconc.m* e *gradfuncoste.m*.
Facemos unha declaración de todas as variables globais no arquivo e realizamos unha asignación de valores: $K = 10\,[m/s]$, $Q = 1{\cdot}10^4\,[kg{\cdot}s^{-1}]$, $u = 10[m/s]$, $H = 2\,[m]$ e $Href = 1.62\,[m]$.

Creamos agora o dominio baixo o que vamos realizar o problema de minimización (3.1) e representamos graficamente a concentración asociada utilizando a función *pinta.m*.

Tomamos $L1 = 10$ e $L_2 = 20$,

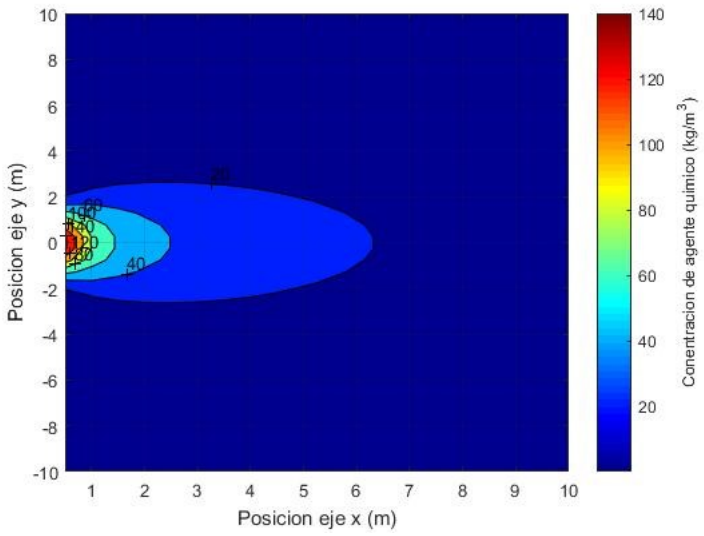


Figura 3.3: Representación gráfica da concentración no dominio Ω, que obtemos do arquivo *pinta.m*.

Coa finalidade de comprobar que o cálculo do gradiente da concentración e o gradiente do funcional de coste son os axeitados, deseñamos dous tests para verificar que ambas funcións están ben definidas.

Sexa $\delta > 0$ e $\vec{e_k} = (0, \dots, 1, \dots, 0)$ o vector n-dimensional da base canónica cuxa compoñente k-ésima é 1:

- **Test 1**

$$\frac{\partial C(x,y)}{\partial x} \simeq \frac{C(x+\delta,y) - C(x,y)}{\delta}, \qquad \frac{\partial C(x,y)}{\partial y} \simeq \frac{C(x,\delta+y) - C(x,y)}{\delta}. \tag{3.31}$$

- **Test 2**

$$\frac{\partial f(\vec{x},\vec{y})}{\partial x_k} \simeq \frac{f(\vec{x}+\delta\vec{e_k},\vec{y}) - f(\vec{x},\vec{y})}{\delta}, \qquad \frac{\partial f(\vec{x},\vec{y})}{\partial y_k} \simeq \frac{f(\vec{x},\vec{y}+\delta\vec{e_k}) - f(\vec{x},\vec{y})}{\delta}, \tag{3.32}$$

  para $k = 2, \ldots, n$.

No arquivo *test1.m* facemos unha comprobación do primeiro test, (3.5), para valores de $\lambda_1 = \lambda_2 = \frac{1}{2}$ sobre o punto $(5,5)$ avaliando en varios valores de δ. Imos denotar o gradiente real por $GCr = \left(\frac{\partial C(x,y)}{\partial x}, \frac{\partial C(x,y)}{\partial y}\right)$ e $GCa = \left(\frac{C(x+\delta,y)-C(x,y)}{\delta}, \frac{C(x,\delta+y)-C(x,y)}{\delta}\right)$ como gradiente aproximado.

Os resultados veñen dados na seguinte táboa,

Delta	$\|Gr - Ga\|_2$
1.e-01	2.152612612501591e-03
1.e-02	2.169513253190479e-04
1.e-03	2.171179221199909e-05
1.e-04	2.171345674829020e-06
1.e-05	2.171367774182412e-07
1.e-06	2.175917506800589e-08
1.e-07	2.861413758775215e-09
1.e-08	5.223957174359925e-09
1.e-09	6.236808882507541e-08
1.e-10	4.820265064733031e-07

Táboa 3.1: Comprobación de que a función concentración está ben definida.

Para o segundo test, (3.32), construímos o arquivo *test2.m* evaluando esta vez para $\lambda_1 = 0.1$, $\lambda_2 = 0.9$ sobre os vectores $\vec{x} = (10,9,8,7,6,5,4,3,2,1)$ e $\vec{y} = (10,10,10,10,10,10,10,10,10,10)$. Denotamos $Gfr = \left(\frac{\partial f(\vec{x},\vec{y})}{\partial x_k}, \frac{\partial f(\vec{x},\vec{y})}{\partial y_k}\right)$ e $Gfa = \left(\frac{f(\vec{x}+\delta\vec{e_k},\vec{y})-f(\vec{x},\vec{y})}{\delta}, \frac{f(\vec{x},\vec{y}+\delta\vec{e_k})-f(\vec{x},\vec{y})}{\delta}\right)$.

Delta	$\|Gfr - Gfa\|_2$
1.e-01	2.655585442152519e-01
1.e-02	2.665359824913051e-02
1.e-03	2.665761669161933e-03
1.e-04	2.665795739694417e-04
1.e-05	2.665781444075070e-05
1.e-06	2.663474792833409e-06
1.e-07	2.602968366441347e-07
1.e-08	2.954298935673284e-07
1.e-09	2.678165838080255e-06
1.e-10	2.917756281254065e-05

Táboa 3.2: Comprobación de que o funcional de coste está ben definido.

Agora ben, como podemos observar na figura 3.1, o punto fonte do vertido atópase no $(0, 0)$, nós imos tomar como punto final da ruta óptima $b = (x_{n+1}, y_{n+1}) = (1, 0)$, que consideramos suficientemente preto do punto fonde como para proceder á súa neutralización, e ademais conservamos tódolos valores xa asignados: $K = 10$, $Q = 1 \cdot 10^4$, $u = 10$, $H = 2$, $Href = 1.62$, $L_1 = 10$ e $L_2 = 20$.

Implementamos o método iterativo para (3.17) na función *grad_ conx_ prox.m* e o proceso do cálculo da proxección ortogonal por casos, tal como observamos previamente na figura 3.2, na función *proxeccion.m*.

No programa principal *main.m* describimos 4 rutas iniciais sobre as que realizaremos diversas probas para comprobar o funcionamento do método iterativo:

- **Proba 1: camiño máis curto**

  Consideramos $\lambda_1 = 0$ e $\lambda_2 = 1$ e sexa $a = (x_0, y_0) = (-10, 10)$ o punto de saída da ruta óptima. Tomamos $\vec{x_0}$=(10, 9.182, 8.364, 7.545, 6.727, 5.909, 5.091, 4.273, 3.454, 2.636,1.818, 1) e $\vec{y_0}$=(-10, -9.090, -8.181, -7.273, -6.364, -5.454, -4.545, -3.636, -2.727, -1.818, -0.909, 0) como iterantes iniciais, que definen unha curva inicial $\sigma_{\vec{x_0},\vec{y_0}}$, que vemos representada na figura 3.4.

  A nosa función de coste asociada neste caso ven dada por:

$$f(\vec{x}, \vec{y}) = \sum_{k=0}^{n} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}. \tag{3.33}$$

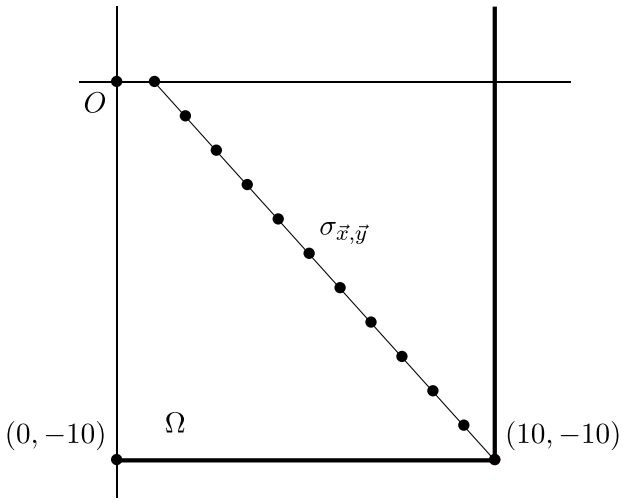


Figura 3.4: Representación gráfica dos vectores x_0 e y_0 que conforman o camiño óptimo da Proba 1.

Executado o método iterativo no programa *main.m* obtemos que na primeira iteración o método da por terminado, como podiamos deducir de forma intuitiva, e verificando así a boa construción do método. Neste caso concreto obtemos un coste asociado de 13.4536, como podemos observar na figura 3.5.

```
*****************************************************
Iteracion 1
Coste asociado 13.4536
Condicion de parada (gradiente) 2.2699e-15
Condicion de parada (iterantes consecutivos) 0
Condicion de parada (función de coste) 0
*****************************************************
```

Figura 3.5: Saída por pantalla do número de iteracións realizadas, do valor do funcional de coste asociado ó camiño óptimo e do valor da condición de parada para a Proba 1 en Matlab.

- **Proba 2: camiño máis longo**

  Consideramos $\lambda_1 = 0$ e $\lambda_2 = 1$, con $a = (-10, 10)$, tendo de novo a mesma función de coste que no test anterior, (3.33).

  Partimos agora de $\vec{x_0} = (8.19, 6.4, 4.6, 2.8, 1, 1, 1, 1, 1, 1)$ e $\vec{y_0} = (-10, -10, -10, -10, -10, -10, -9, -7.2, -5.4, -3.6, -1.8)$ cuxa curva asociada $\sigma_{\vec{x_0},\vec{y_0}}$ ven representada na figura 3.6.

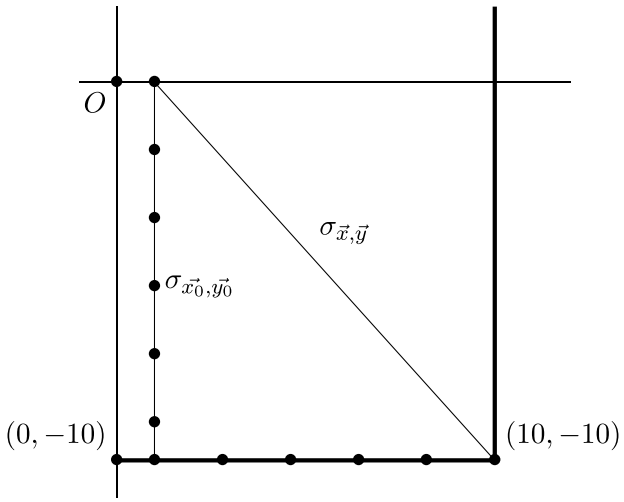


Figura 3.6: Representación gráfica da Proba 2 considerando só o gradiente como condición de parada. Sendo $\sigma_{\vec{x_0},\vec{y_0}}$ o camiño inicial a minimizar, e $\sigma_{\vec{x},\vec{y}}$ o camiño óptimo.

Imos considerar primeiro un único test de parada, o gradiente, e vemos que neste test o camiño óptimo é o mesmo que na proba1, e polo tanto o funcional de coste toma o mesmo valor. Para esta proba o método precisa de un número superior de interacións ata chegar ó camiño óptimo, como podemos observar na figura 3.7.

```
*****************************************************
Iteracion 1760
Coste asociado 13.4536
Condicion de parada (gradiente) 9.9895e-05
*****************************************************
```

Figura 3.7: Saída por pantalla da Proba 2 en Matlab só considerando o gradiente como condición de parada.

Se agora consideramos todos os tests de parada descritos na sección 3.2.1 (gradiente nulo, diferencia de iterantes consecutivos e diferencia de coste) obtemos unha ruta óptima distinta, fig. 3.8. Isto débese a que consideramos unha dirección de descenso $d = 0.01$ e unha tolerancia $tol = 1 \cdot 10^{-4}$. Se aumentamos, por exemplo, a dirección de descenso en $d = 0.2$ e tomamos $tol = 1 \cdot 10^{-5}$, o proceso iterativo resolve o problema en moitas menos iteracións e acadando a mesma ruta óptima que na fig. 3.6.

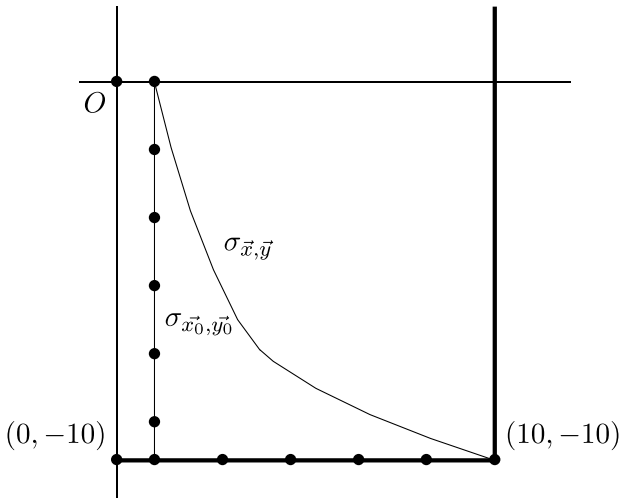


Figura 3.8: Representación gráfica da Proba 2 xerada en Matlab , considerando todos os tests de parada.

Neste caso, o proceso iterativo se interrompe antes debido a que se verifica a condición de parada asociada a diferencia de función de costes entre iterantes consecutivos, antes que calquera das outras dúas. Isto deixa un coste asociado maior que o que se obtiña utilizando unha única condición de parada, fig. 3.9. Cando se trata de un caso real habería que entrar a valorar cal dos camiños compensa.

```
*****************************************************
Iteracion 1482
Coste asociado 14.5667
Condicion de parada (gradiente) 0.92879
Condicion de parada (iterantes consecutivos) 0.0054544
Condicion de parada (función de coste) 6.8735e-05
*****************************************************
```

Figura 3.9: Saída por pantalla da Proba 2 en Matlab considerando todos os tests de parada.

- **Proba 3**

Consideramos $\lambda_1 = \lambda_2 = \frac{1}{2}$, para $a = (0, 10)$ e $\vec{x_0} = (10, 8.4, 7.3, 6.4, 5.5, 4.6, 3.74, 2.9, 2.23, 1.7, 1.3, 1)$ e $\vec{y_0} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. A nova función de coste asociada ven dada por:

$$f(\vec{w}) = \frac{1}{2} \cdot g(\vec{w}) + \frac{1}{2} \cdot \sum_{k=0}^{n} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}. \qquad (3.34)$$

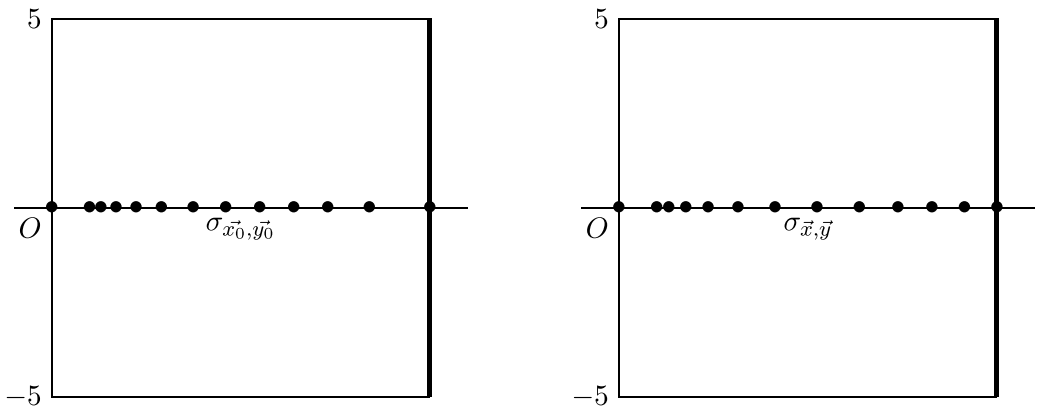


Figura 3.10: Representación gráfica do camiño inicial (drch.) e o camiño óptimo (izq.) da Proba 2.

Para esta proba imos obter un camiño inicial e final totalmente indénticos, fig. 3.10, pero sen embargo o proceso iterativo realiza un total de 1324 interaccións ata chegar ó que considera a ruta óptima, como podemos apreciar na figura 3.11.

Estes resultados débense aos erros de precisión que se producen cando aproximamos a integral da concentración mediante a regra do trapecio simple (3.12), tema que trataremos con máis detalle no capítulo seguinte.

```
*******************************************************
Iteracion 1324
Coste asociado 128.735
Condicion de parada (gradiente) 0.099931
Condicion de parada (iterantes consecutivos) 0.00099972
Condicion de parada (funcion de coste) 9.9923e-05
*******************************************************
```

Figura 3.11: Saída por pantalla da Proba 3 en Matlab considerando todos os tests de parada.

- **Proba 4**

  Vexamos que sucede co noso método iterativo cando consideramos $\lambda_1 = 1$ e $\lambda_2 = 0$. Tomamos de novo o punto de partida $(10, -10)$ e o camiño máis curto que definimos na Proba 1.

  O funcional de coste para esta proba depende unicamente da concentración:

$$f(\vec{x},\vec{y}) = g(\vec{x},\vec{y}) = \sum_{k=0}^{n} \sqrt{(x_{k+1}-x_k)^2 + (y_{k+1}-y_k)^2}\frac{1}{2}\left[C\left(x_k,y_k\right) + C\left(x_{k+1},y_{k+1}\right)\right]. \tag{3.35}$$

```
*****************************************************
Iteracion 100000
Coste asociado 135.0721
Condicion de parada (gradiente) 29.5423
Condicion de parada (iterantes consecutivos) 0.25429
Condicion de parada (función de coste) 0.75565
*****************************************************
```

Figura 3.12: Saída por pantalla da Proba 4 en Matlab.

Neste caso o noso algoritmo non é capaz de chegar a unha solución óptima en 100.000 iteracións posto que ningunha das condicións de parada se verifica, fig. 3.12.

Nas probas anteriores xa puidemos comprobar que a pesaren de obter unha ruta óptima o método iterativo precisa de numerosas iteracións para chegar a unha solución.

3.3.1. Regra de Armijo

En vista da lentitude de converxencia do método imos considerar un cambio na elección do paso de descenso.

No método co que estamos traballando o paso de descenso d viña dado como unha constante calqueira positiva, agora utilizaremos a *Regra de Armijo* [3] para facer unha selección de paso máis precisa.

Fixamos os escalares[4] δ, γ e d^0. Esta regra está baseada na redución sucesiva do paso, escollemos un paso inicial e se non se verifica que

$$f\left(P_\Omega((x^k, y^k) - d^0 \nabla f(x^k, y^k))\right) \leq f(x^k, y^k) - \gamma d^0 \|\nabla f(x^k, y^k)\|^2, \tag{3.36}$$

o paso redúcese, $d^k = d^{k-1}\delta$. Este proceso se repite ata que o método converxe a unha solución óptima.

Realizamos de novo as probas descritas no capítulo anterior sobre o novo programa e comparamos entre ambos métodos na seguinte táboa:

[3] A nova elección de paso está descrita no arquivo *Armijo.m* que atopamos no Anexo.

[4]

	Proba 2		Proba 3		Proba 4	
	Primitivo	*Armijo*	Primitivo	*Armijo*	Primitivo	*Armijo*
Iteración	457	314	1324	675	nitmax	38
Tempo	2.25 s	2.22 s	6.678 s	4.11 s	421.2686 s	0.5698 s
Coste	13.454	13.458	128.735	110.13	135.0721	90.0124

Observamos que claramente a nova elección do paso mellora o método de forma moi eficiente.

3.3.2. A función *fmincon*

Visto que o noso algoritmo ten algunhas deficiencias con respecto aos tempos de converxencia, imos considerar outras alternativas para a resolución do problema de minimización sobre o que estamos traballando.

O propio programa Matlab posúe ferramentas para resolución de problemas de minimización. No noso caso a que máis se axusta ao noso problema é a función *fmincon.m*, e coa que faremos unha pequena comparación.

A función *fmincon* ten as seguintes variables de entrada:

fun: Función obxectivo que queremos minimizar.

x0: Vector semilla.

A: Matriz asociada as restricións de desigualdade.

b: Segundo membro asociado as restricións de desigualdade.

Aeq: Matriz asociada as restricións de igualdade.

beq: Segundo membro asociado as restricións lineais.

lb: Cotas inferiores ás que está restrinxida a solución.

ub: Cotas inferiores ás que está restrinxida a solución.

nonlcon: Restricións non lineais de desigualdade.

options: Diversas opcións da propia fmincon que permiten a elección do algoritmo de minimización.

Está preparado para resolver problemas de minimización con restricións de varios tipos,

$A \cdot x \leq B$ restrición lineal de desigualdade,

$Aeq \cdot x = beq$ restrición lineal de igualdade,

$lb \leq x \leq ub$ acotación superior e inferior da solución.

En *options* temos a posibilidade de escoller entre varios algoritmos para realizar o proceso de minimización:

interior-point: Algoritmo de puntos interiores

trust-region-reflective: Algoritmo de confianza-rexión-reflexivo

sqp: Programación cadrática secuencial

active-set: Algoritmo do conxunto activo

Exliquemos brevemente o funcionamento dos algoritmos [9]:

- **Algoritmo de Puntos Interiores**

  Para un problema de minimización tal que:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{suxeito a} \quad & h(x) = 0 \\ & g(x) \leq 0, \end{aligned} \tag{3.37}$$

  construímos un problema aproximado

$$\begin{aligned} \min_{x,\,s} \quad & f_\mu(x,s) = \min_{x,s} f(x) \; - \; \mu \sum_i ln(s_i) \\ \text{suxeito a} \quad & h(x) = 0 \\ & g(x) \; + \; s = 0 \\ & s_i > 0 \quad \forall\, i. \end{aligned} \tag{3.38}$$

  Como μ tende a cero o mínimo da función aproximada f_μ debe aproximarse ao mínimo de f. Definimos $\sum_i ln(s_i)$ como a función barreira.

  O algoritmo intenta en cada iteración a diminución da que se define como "función de mérito",

$$f_\mu(x,s) + \nu \|(h(x), g(x) + s)\|. \tag{3.39}$$

  Utiliza dous pasos distintos para este proceso, un por defecto, paso directo, e outro para cando non se produce unha diminución da función de mérito, paso do gradiente conxugado. O algoritmo tamén rechaza unha solución que devolva un *Nan* ou un *INF*.

- **Algoritmo de confianza-rexión-reflexivo**

  Este tipo de método está baseado no concepto de rexión de confianza. A idea principal consiste en aproximar a función f a optimizar por unha función máis simple, q, que describa o comportamento de f nun entorno de x, sendo este entorno o que se coñece por rexión de confianza.

Esta idea deriva nun problema de minimización da rexión de confianza, N, é dicir

$$\min_s \{q(s),\ s \in N\}. \tag{3.40}$$

A función *fmincon* define a rexión de confianza tal que

$$\min \left\{ \frac{1}{2} s^T H s + s^T \nabla f,\ \|Ds\| \leq \Delta \right\}, \tag{3.41}$$

sendo H a matriz Hessiana, D unha matriz de escalamento diagonal e Δ un escalar positivo que determina a dimensión da rexión de confianza.

O proceso iterativo se actualiza cando $f(x+s) < f(x)$, se non, reducimos a rexión de confianza reaxustando Δ.

- **Algoritmo do conxunto activo**

Neste algoritmo partimos de un problema de optimización tal que

$$\begin{aligned} \min_x \quad & f(x) \\ \text{suxeito a} \quad & G_i(x) = 0, \quad i = 1, \ldots, m_e \\ & G_i(x) \leq 0, \quad i = m_e + 1, \ldots, m, \end{aligned} \tag{3.42}$$

sendo $G(x)$ unha función vectorial que devolve as restricións. Engadimos a maiores como condicións necesarias para a optimización as coñecidas como ecuacións KKT (*Karush-Kuhn-Tucker*):

$$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \cdot \nabla G_i(x) = 0 \tag{3.43}$$

$$\lambda_i \cdot G_i(x) = 0, \quad i = 1, \ldots, m_e \tag{3.44}$$

$$\lambda_i \leq 0, \quad i = m_e + 1, \ldots, m, \tag{3.45}$$

sendo λ_i os multiplicadores de Lagrange.

A idea principal de este algoritmo é a formulación dun subproblema de programación cadrática baseado nunha aproximación cadrática da función de Lagrange,

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot \nabla_i f(x). \tag{3.46}$$

Este subproblema ven dado como:

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \frac{1}{2} d^T H d + (f(x_k))^T d \\ \text{suxeito a} \quad & (\nabla_i f(x_k))^T d + \nabla_i f(x_k) = 0 \quad i = 1, \ldots, m_e \\ & (\nabla_i f(x_k))^T d + \nabla_i f(x_k) \leq 0 \quad i = m_e + 1, \ldots, m. \end{aligned} \tag{3.47}$$

- **Algoritmo *sqp***

  Este algoritmo segue a mesma estrutura que o algoritmo *active-set* pero alberga unhas pequenas diferencias:

  1. Na función *fmincon* o algoritmo *sqp* pode intentar un paso que "falle". Antes de obter un *INF*, un *Nan* ou valor complexo o algoritmo intenta dar un paso máis pequeno.
  2. Rutinas de álxebra lineal refactorizadas: son máis eficientes en memoria e en velocidade.
  3. Rutinas de viabilidade reformuladas: poden ralentizar a solución ao requirir máis evolucións das funcións de restrición non lineais.

O algoritmo *interior-point* é o que ven por defecto e é o que usaremos neste traballo posto que da mellores resultados. Cabe destacar que o resto de algoritmos funcionan sobre o problema de minimización que intentamos resolver, pero algúns como o *sqp* ou *active-set* chegan á solución moi lentamente ou devolven camiños de dubidosa utilidade. Todo isto se comentará de con máis detalle máis adiante neste propio capítulo e no seguinte.

Procedemos agora a realizar co programa *fminconp.m* as mesmas probas que para o noso algoritmo, e obtemos:

- **Proba 1**

  Para $\lambda_1 = 0$ e $\lambda_2 = 1$, partindo de novo desde o camiño máis curto entre o punto final $(1, 0)$ e o punto de inicio $(10, -10)$.

```
Optimization completed because at the initial point, the objective function is non-decreasing
in feasible directions to within the default value of the optimality tolerance, and
constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>

                                          First-order      Norm of
 Iter F-count            f(x)  Feasibility  optimality         step
    0       1    1.345362e+01    0.000e+00    7.414e-16

Initial point is a local minimum that satisfies the constraints.
```

Figura 3.13: Saída por pantalla da Proba 1 en Matlab utilizando a función fmincon.

Como cabería esperar, na primeira iteración o algoritmo da por finalizado o proceso determinando que o camiño inicial é o óptimo. Observamos tamén que o coste asociado é o mesmo que obtivemos co anterior algoritmo.

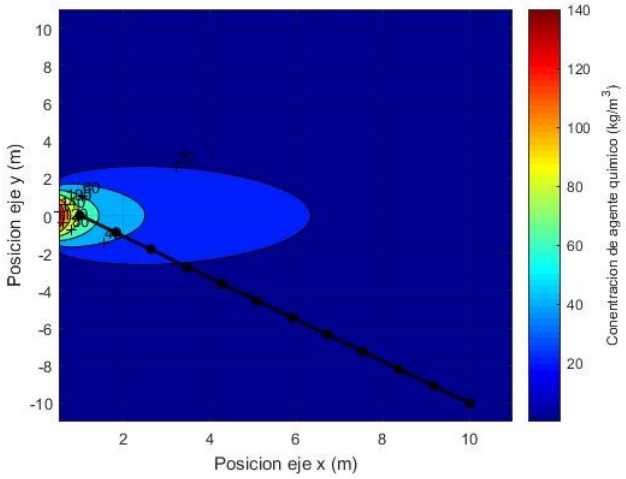
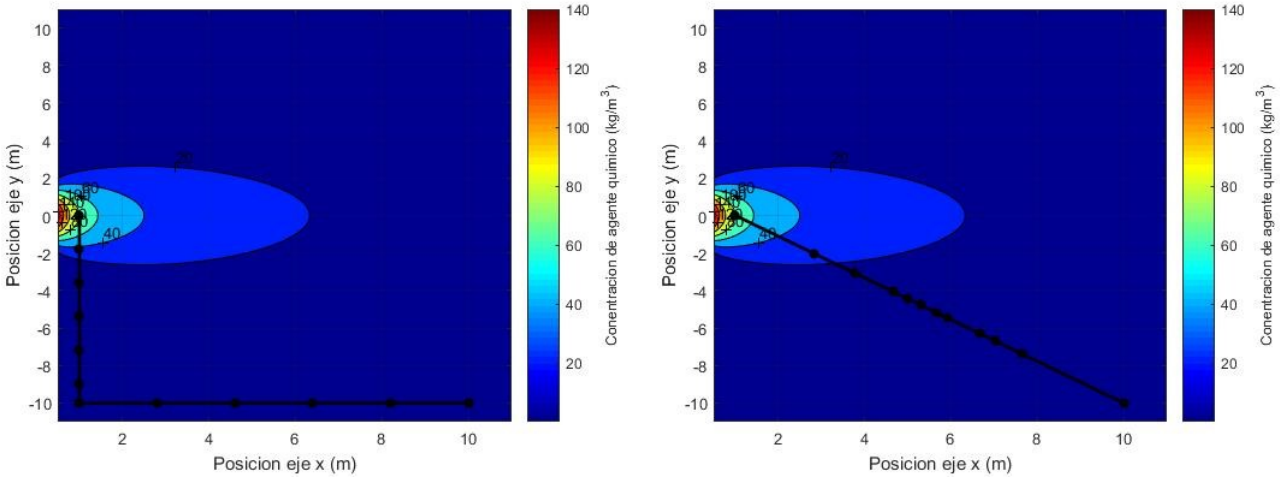


Figura 3.14: Representación da ruta óptima para a Proba 1 sobre o gráfico de concentracións.

- **Proba 2**

  Tomando os mesmos lambdas da Proba 1, e partindo esta vez do camiño máis longo entre $(1, 0)$ e $(10, -10)$, obtemos a mesma solución que co algoritmo desenrolado anteriormente.



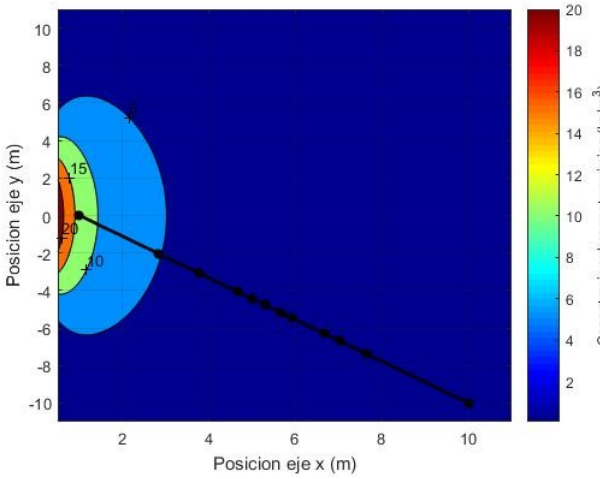
(a) Camiño inicial (b) Camiño óptimo

Figura 3.15: Representación gráfica da Proba 2.

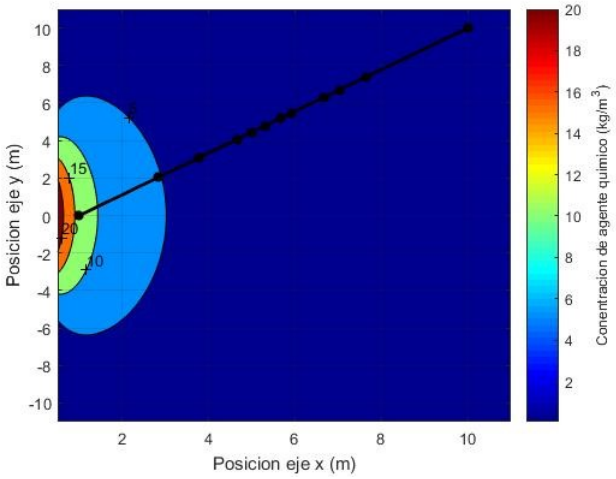
Para estas dúas primeiras probas o problema de optimización da ruta óptima redúcese a un problema de minimización de distancia que só vai depender dos valores que tomemos

para o punto de saída $a = (x_0, y_0)$ e o punto final do camiño $b = (x_n, y_n)$, sendo a ruta óptima a recta que une ambos puntos, independentemente da elección do camiño inicial,

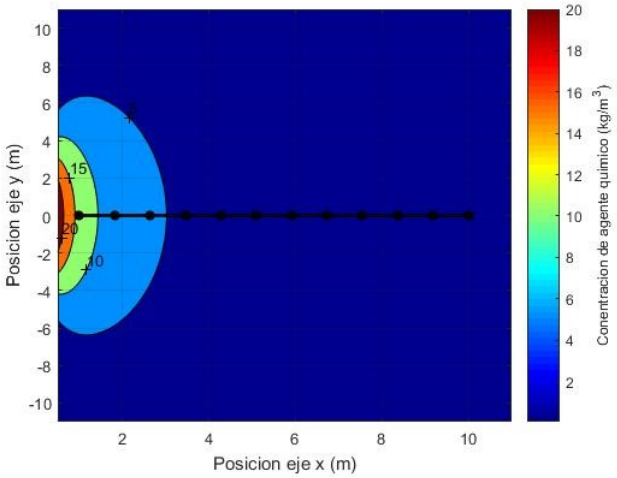
$$\sigma_{\vec{x},\vec{y}}(t) = (x_0, y_0) + t((x_n, y_n) - (x_0, y_0)), \quad t \in [0, 1]. \tag{3.48}$$



(a) Saíndo desde o $(10, -10)$.



(b) Saíndo desde o $(10, 10)$.



(c) Saíndo desde o $(10, 0)$.

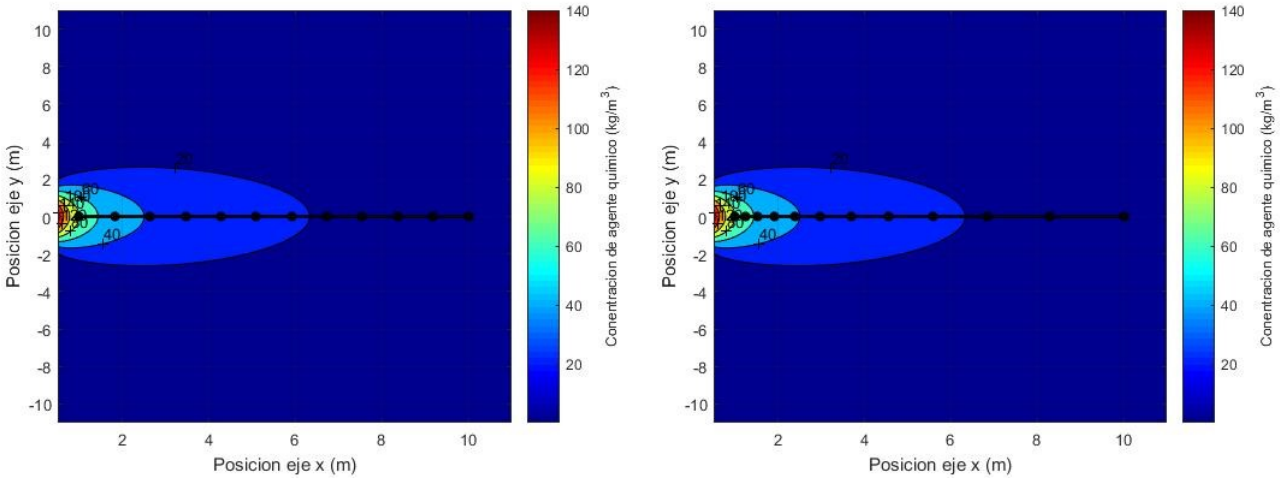
Figura 3.16: Representación gráfica do camiño óptimo para $\lambda_1 = 0$ e $\lambda_2 = 1$ con $(1, 0)$ punto final e $(0, 0)$ punto fonte. Con $K = 1 \cdot 10^2$.

- **Proba 3**

  Con $\lambda_1 = \lambda_2 = \frac{1}{2}$ e tomando o camiño máis curto entre o punto $(1, 0)$ e $(10, 0)$.

  Igual que co outro algoritmo, debido ós problemas de precisión que provoca a aproximación da integral da concentración, se realizan varias iteracións para atopar unha "solución óptima".

  De novo observamos que o propio camiño incial é o óptimo nesta proba.



(a) Camiño inicial (b) Camiño óptimo

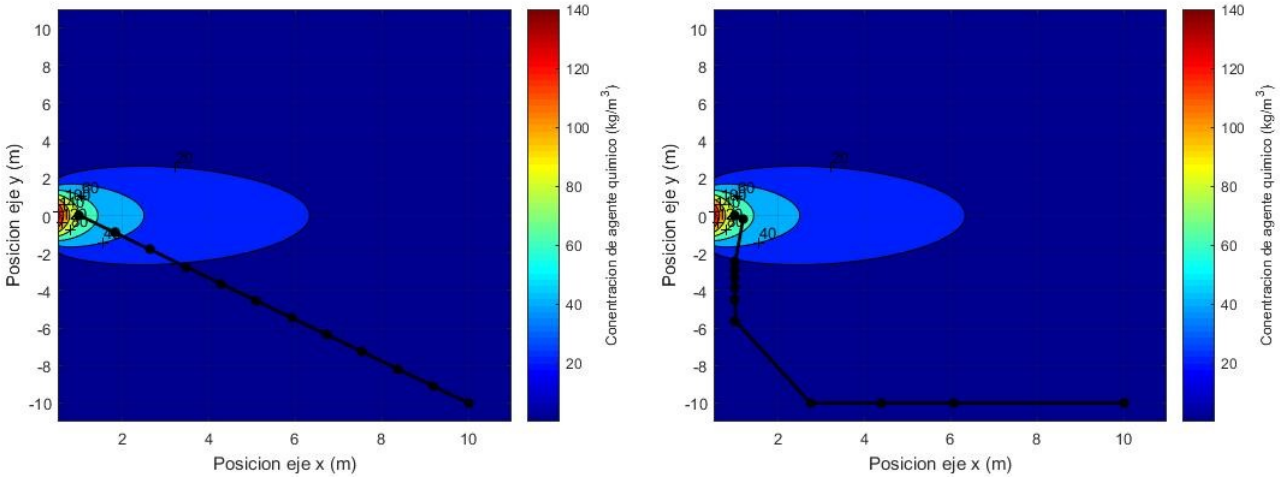
Figura 3.17: Representación gráfica da Proba 3.

- **Proba 4**

  Tomamos $\lambda_1 = 1$ e $\lambda_2 = 0$ para o camiño incial da Proba 1.

  Ao contrario que co anterior algoritmo, a función *fmincon* é capaz de atopar unha solución en 85 iteracións, como podemos observar na figura 3.18.

  O camiño óptimo ten asociado neste caso un funcional de coste de 121.8172.



(a) Camiño inicial (b) Camiño óptimo

Figura 3.18: Representación gráfica da Proba 4.

Podemos realizar esta mesma proba partindo desde puntos do dominio que consideramos significativos, e así analizar con máis detalle a elección de ruta óptima que fai o algoritmo.

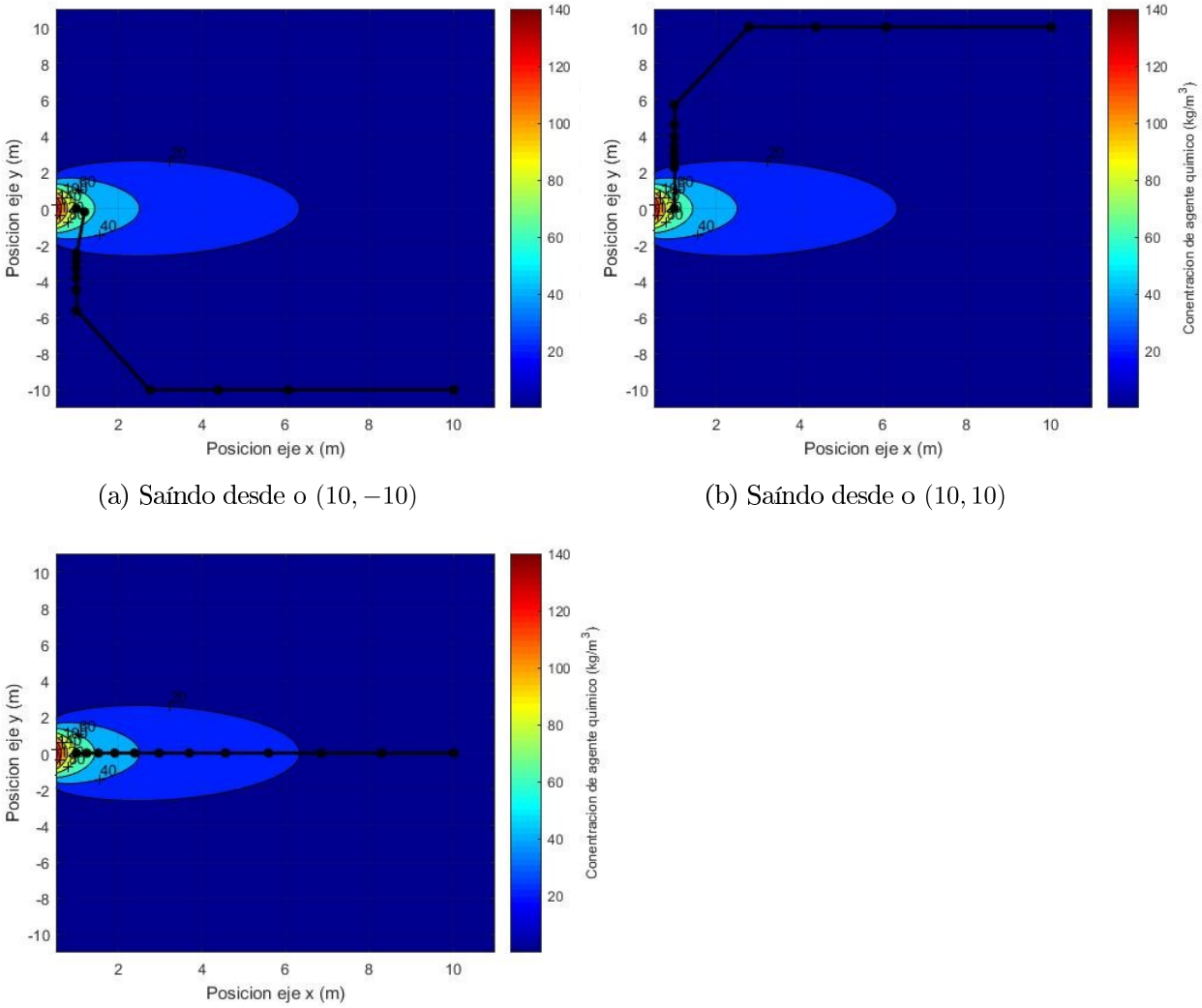
Lembremos que partimos dos valores:$K = 10\,[m/s]$, $Q = 1 \cdot 10^4\,[kg \cdot s^{-1}]$, $u = 10[m/s]$, $H = 2\,[m]$, $Href = 1.62\,[m]$, e $n = 10$.

Nesta alternativa temos unha nube de concentración na forma máis estándar de campana de Gauss sobre o dominio.

Analizando o punto de saída, (x_0, y_0), sen ter en conta o camiño inicial, observamos que se o punto inicial está por debaixo do punto fonte o método escolle un camiño por debaixo da nube de contaminación e á altura da posición do punto fonte redirixe a ruta directamente ata arriba e así minimizar a exposición ao axente químico. Ocorre de forma análoga se o punto inicial está por enriba do punto fonte.

Sen embargo, se saímos desde un punto á mesma altura que o punto fonte a ruta óptima vai ser a recta que une o punto de saída co punto fonte. De novo estamos ante os problemas de aproximación que comentaremos no capítulo seguinte.

Comparando os costes asociados que obtemos partindo desde distintos puntos do dominio se observa que saíndo desde a mesma altura o coste e moito maior, $f(\vec{x}, \vec{y}) = 248.087$, ao contrario que se variamos a posición, tanto por enriba como por debaixo, $f(\vec{x}, \vec{y}) = 134.653$.



(a) Saíndo desde o $(10, -10)$

(b) Saíndo desde o $(10, 10)$

(c) Saíndo desde o $(10, 0)$

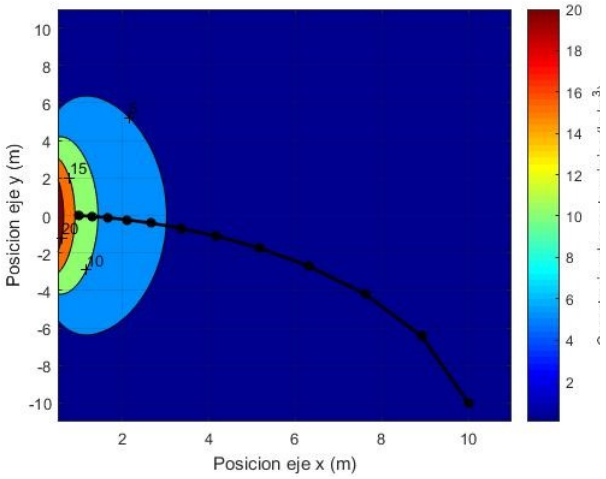
Figura 3.19: Representación gráfica do camiño óptimo para $\lambda_1 = 1$ e $\lambda_2 = 0$ con $(1, 0)$ punto final e $(0, 0)$ punto fonte. Con $K = 10$.

Modificamos agora o valor da constante de difusión que pasará a ser $K = 1 \cdot 10^2$. Isto provoca un ancheamento da campá de Gauss sobre o dominio.

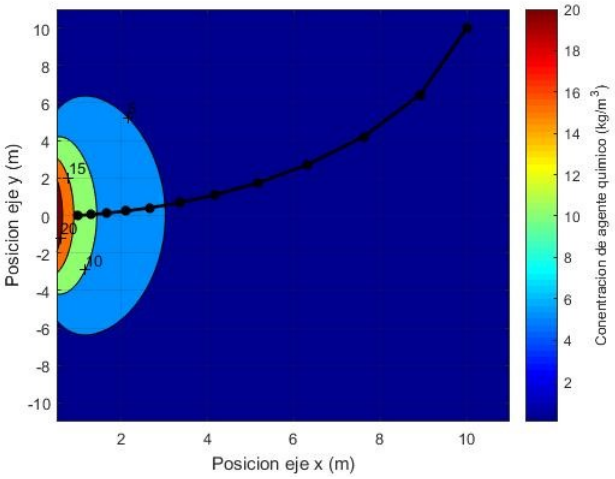
Neste caso saíndo desde un punto por enriba ou por debaixo do punto fonte o camiño óptimo resulta ser aquel que intenta entrar o máis de fronte posible á nube de concentración. Por iso cando partimos desde a mesma altura entra totalmente de fronte.

Ao contrario que para $K = 10$, agora o coste é menor se partimos desde a mesma

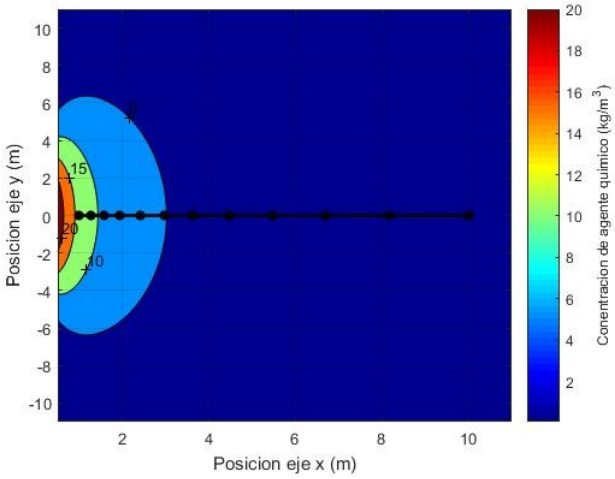
altura, $f(\vec{x},\vec{y}) = 34.69$, que se partimos desde calquera das outras, $f(\vec{x},\vec{y}) = 43.98$.



(a) Saíndo desde o $(10,-10)$.



(b) Saíndo desde o $(10,10)$.



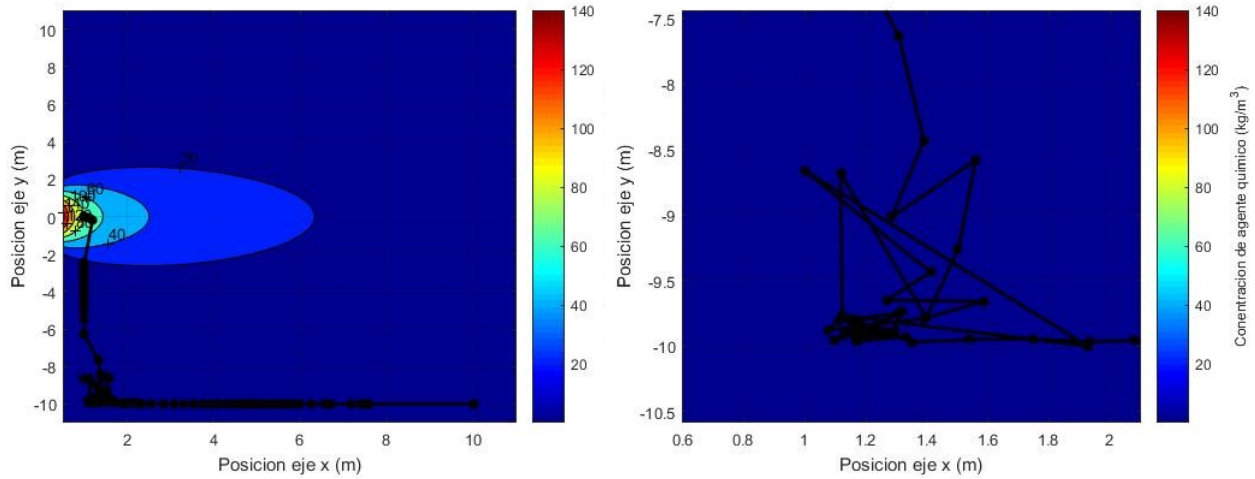
(c) Saíndo desde o $(10,0)$.

Figura 3.20: Representación gráfica do camiño óptimo para $\lambda_1 = 1$ e $\lambda_2 = 0$ con $(1,0)$ punto final e $(0,0)$ punto fonte. Con $K = 1 \cdot 10^2$.

Con respecto a Proba 4 debemos destacar que a función *fmincon* non pode xestionar o aumento do número de puntos que utilizamos no método iterativo. Imos ver un exemplo claro de este problema a continuación.

Para $\lambda_1 = 1$ e $\lambda_2 = 0$ na proba 4, se utilizamos un número de nodos moi alto, $n = 100$, obtemos unha solución estraña, figuras 3.21 e 3.22, posto que ao só facer depender a

función de coste do valor de concentración, temos de novo os problemas de precisión pola aproximacion do trapecio simple. Así pois, para poder resolver o problema de minimización utilizando *fmincon* debemos facer unha redución de nodos considerable se queremos obter unha solución óptima con sentido.
Poderiamos considerar que a solución obtida polo algoritmo *trust-region-reflective* é "válida", pero temos que ter en conta que o proceso de cálculo realizouse en $855.778\,s$, un tempo superior ao que precisaron o resto de algoritmos para atopar unha "ruta óptima". Destacamos tamén que o algoritmo *active-set* non foi quen de acadar unha solución, posto que a metade de proceso devolve valores de tipo *NaN*.



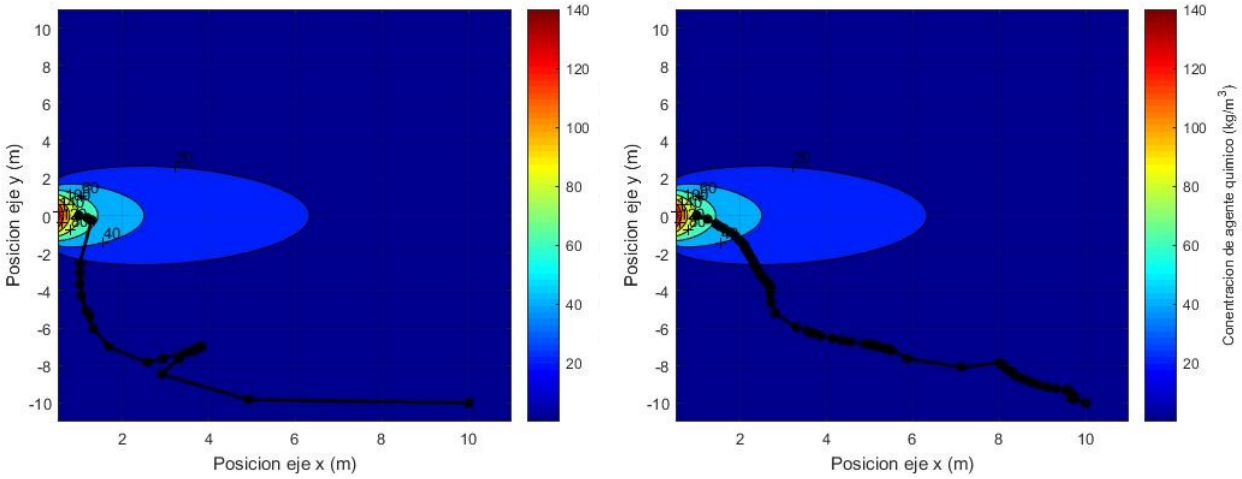
(a) Algoritmo *sqp*.

(b) Zoom sobre o a ruta obtida polo algorimo *sqp*.

Figura 3.21: Representación gráfica xerada por Matlab da Proba 4 utilizando *fmincon*. Para 100 nodos e $\lambda_1 = 1$, $\lambda_2 = 0$ utilizando o algoritmo *sqp*.

Por outra banda, conservando o número de nodos, se modificamos o valor das lambdas de tal forma que para λ_1 o seu valor estea preto de 1, obtemos que atopa unha solución óptima de forma rápida para o método *sqp*, e que atopa unha ruta moi semellante á anterior para o algoritmo*trust-region-reflective*, como podemos observar na figura 3.23. O método de puntos interiores continúa obtendo resultados estraños e o algoritmo *active-set* segue sen poder chegar a unha ruta óptima.

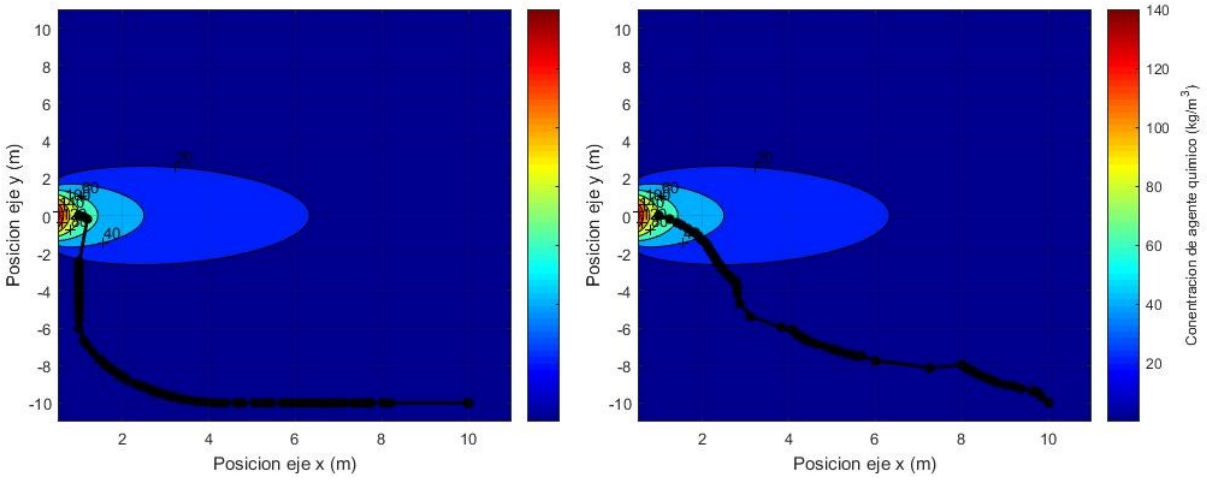
En vista destos resultados podemos seleccionar o algoritmo *trust-region-reflective* como o máis indicado neste exemplo. Pero consideramos que ante un alto número de nodos, sempre que ningún dos lambdas alcance o valor máximo de 1, o algoritmo *sqp* responde de mellor forma que ningún outro.



(a) Algorimo de puntos interiores.

(b) Algoritmo de confianza-rexión-reflexivo.

Figura 3.22: Representación gráfica xerada por Matlab da Proba 4 utilizando *fmincon*. Para 100 nodos e $\lambda_1 = 1$, $\lambda_2 = 0$.



(a) Algoritmo *sqp*. Coste asociado de 121.35 e tempo de cáculo, 14.25 s.

(b) Algoritmo *trust-region-reflective*. Coste asociado de 159.613 e tempo de cálculo, 837.22 s.

Figura 3.23: Representación gráfica xerada por Matlab da Proba 4 utilizando *fmincon*. Para 100 nodos e $\lambda_1 = 0.9$, $\lambda_2 = 0.1$.

3.3.3. Comparación de Métodos

Podemos facer unha táboa comparativa bastante esclarecedora sobre que método numérico dos que se trataron no Capítulo 3 é moito máis útil para a resolución do noso problema de minimización, baseándonos nos resultados obtidos nas probas correspondentes.

Lembramos que estamos realizando as probas tomando $K = 10\,[m/s]$, $Q = 1 \cdot 10^4\,[kg \cdot s^{-1}]$, $u = 10[m/s]$, $H = 2\,[m]$, $Href = 1.62\,[m]$, e $n = 10$.

	Proba 2		Proba 3		Proba 4	
	Primitivo*1	*fmincon*	Primitivo	*fmincon*	Primitivo*2	*fmincon*
Iteración	457	24	1324	42	nitmax	70
Tempo	2.25 s	1.14 s	6.678 s	3.18 s	421.2686 s	1.13 s
Coste	13.454	13.454	128.735	128.544	135.0721	134.658

Táboa 3.3

*1 Debemos ter en conta as modificacións no valor do paso e da tolerancia que explicamos na sección anterior.

*2 Resultados obtidos para $nitmax = 100000$ que non se corresponden coa solución óptima.

Analizaremos estes resultados no capítulo seguinte.

Capítulo 4

Conclusións

Neste capítulo resumiremos os resultados que obtivemos, poñendo en valor aquelas cuestións que, ao noso xuízo, supoñen aportes positivos e sendo críticos con aquelas cuestións non tan boas. No caso relativo ás deficiencias se expoñerán todos os problemas que nos atopamos despois de realizar as probas numéricas tanto do noso algoritmo como do algoritmo proporcionado por MATLAB. Expoñeremos os posibles motivos polos que se producen e daremos algunha solución factible.

4.1. Resultados

Lembremos que o método do gradiente conxugado para o noso problema de minimización viña dado por

$$(\vec{x'}^{\,n+1}, \vec{y'}^{\,n+1}) = P_\Omega \left((\vec{x'}^{\,n}, \vec{y'}^{\,n}) - d\nabla f(\vec{x}^{\,n}, \vec{y}^{\,n}) \right), \tag{4.1}$$

con $\vec{x'} = (x_1, \dots, x_n)$ e $\vec{y'} = (y_1, \dots, y_n)$. Para d paso fixo, $d \in \mathbb{R}^+$.

Debemos destacar ante todo que o algoritmo primitivo desde a súa simplicidade pode chegar a competir cos resultados obtidos con $fmincon$ aínda que, como puidemos observar na táboa 3.3, para casos máis complexos queda un paso por detrás. A función $fmincon$ non só realiza o proceso iterativo en moitos menos pasos, senón que ademais iguala ou mellora o coste asociado á ruta óptima en cada unha das probas.

Sen embargo, cando realizamos o cambio de elección do paso mediante a Regra de Armijo o método converxe con máis rapidez, e mesmo "supera", hipoteticamente, os resultados obtidos pola función *fmincon*, como podemos observar na seguinte táboa.

	Proba 2		Proba 3		Proba 4	
	Armijo	*fmincon*	*Armijo*	*fmincon*	*Armijo*	*fmincon*
Iteración	314	24	675	42	38	70
Tempo	2.22 s	1.14 s	4.11 s	3.18 s	0.5698 s	1.13 s
Coste	13.458	13.454	110.13	128.544	90.012	134.658

Táboa 4.1

Con respecto aos resultados obtidos empregando a función *fmincon* se consideramos que a función de coste depende tanto da contaminación como da distancia total percorrida, i.e $\lambda_1 \neq 0$ e $\lambda_2 \neq 0$, e analizando as figuras 3.19 e 3.20 podemos deducir o seguinte.
Canto máis ancho sexa o "penacho"de contaminación, fig. 3.20, máis compensa tomar un punto de saída á mesma altura do punto fonte. Mentres que se temos o caso contrario, fig. 3.19, o camiño óptimo será aquel que parta dunha altura distinta do punto fonte e rodee o máximo tempo posible a nube de contaminación antes da súa entrada.

4.2. Problemas básicos do Algoritmo Primitivo

No proceso de construción do método de resolución os criterios de parada que escollimos son os máis simples que poderiamos tomar. Esto inflúe directamente no número de iteracións que precisa o método para converxer a unha solución óptima.

Como puidemos comprobar nas probas realizas no capítulo anterior sobre o algoritmo, este precisa como mínimo un total de 1000 iteracións antes de acadar a ruta final. Destacaremos sobre todo a Proba 4, para a cal non foi quen de acadar unha solución antes das 100.000 iteracións. Podemos considerar varias maneiras posibles para solucionalo: cambiar a elección do paso (empregando a Regra de Armijo), aumentar o número de puntos ou mellorar a fórmula de cuadratura. Analicemos máis a fondo a última opción.

No Capítulo 3, no proceso de cálculo da función de coste, realizamos unha aproximación mediante a Regra do Trapecio Simple, (3.12). Isto ocasiona uns pequenos erros de aproximación que podemos apreciar na Proba 2 para ambos algoritmos, tanto no primitivo, como na función *fmincon*. Observamos que o algoritmo realiza o proceso iterativo ininterrompidamente a pesaren de que a ruta inicial e a óptima son a mesma.

A Regra do Trapecio Simple, como xa vimos, consiste en aproximar o valor dunha

integral definida da seguinte forma.
Sexa f función definida nun intervalo $[a, b]$, a integral $\int_a^b f$ pode aproximarse por $\int_a^b \left[f(a) + \frac{f(b)-f(a)}{b-a}(x-a) \right] dx$, de tal forma que

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}. \tag{4.2}$$

Para que a Regra do Trapecio Simple non cometa ningún erro de aproximación a función f debería ser lineal. No noso caso f é a función concentración, $C(x, y)$ (3.7), que claramente non é lineal[1].

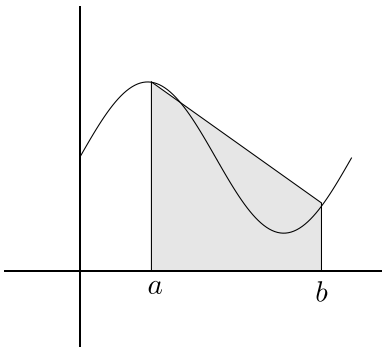


Figura 4.1: Representación gráfica da aproximación da área baixo a curva mediante a Regra do Trapecio Simple

Para que a aproximación da integral sexa máis precisa poderiamos mellorar a fórmula de cuadratura (trapecios composto) ou considerar outras opcións a partir das fórmulas de Newton-Cotes[19]:

- Regra de Simpson.

$$\int_a^b f(x)dx \approx \frac{b-a}{6}[f(a) + 4f(m) + f(b)], \tag{4.3}$$

  con $m = \frac{a+b}{2}$.

- Regra de Newton

$$\int_a^b f(x)dx \approx \frac{b-a}{8}\left[f(a) + 3f\left(\frac{2a+b}{3}\right) + f\left(\frac{a+2b}{3}\right) + f(b) \right]. \tag{4.4}$$

[1]Unha función real f dise lineal se pode expresarse como: $f(x) = mx + b$, con $m, b \in \mathbb{R}$

4.3. Problemas coa funcion *fmincon*: resultados erróneos

A función *fmincon* proporcionada por MATLAB para a resolución de problemas de minimización non está exenta de pequenas deficiencias no momento de cálculo da solución óptima para o noso problema da ruta óptima. Debemos ter en conta que os algoritmos teñen por defecto un número máximo de iteracións e un máximo de puntos que pode avaliar sobre a función obxectivo, estes parámetros terán que ser os apropiados para poder acadar unha solución do problema de optimización.

Neste caso non podemos modificar nada relativo ó algoritmo iterativo, polo que as solucións pasan unicamente por modificar os datos de partida, como a variación do número de puntos, nodos... Non podemos esquecer que, de novo, a función vese afectada polos erros de aproximación da fórmula de cuadratura escollida.

Lembremos que no caso de que a función de coste dependa altamente da concentración, se consideramos un alto número de nodos, o algoritmo delvolve unha solución pouco realista, fig. 3.21b. A redución do número de nodos limita a precisión coa que se pode obter o camiño, polo que fai de esta deficiencia un aspecto a ter en conta antes da elección de *fmincon* como método de resolución.

Capítulo 5

Liñas futuras

Poñendo o punto de mira en como actualizar e mellorar o método de determinación da ruta óptima para afinar a súa precisión, e os tempos de converxencia, expoñeremos neste capítulo varias alternativas que se poden examinar.

En primeira instancia debe considerarse mellorar a aproximación en dúas dimensións do problema tratado neste traballo. Facendo uso de distintas fórmulas de cuadratura para afinar na aproximación da integral da concentración e definindo quizais unha parametrización distinta da curva que tomamos como ruta inicial.

O modelo *Gaussian Plume* é o que se considera un modelo estacionario, xa que estimamos que o punto fonte emite de forma continua e a unha velocidade constante, entón poderiamos apreciar tamén a posibilidade substituír este modelo por outro de tipo evolutivo, como o *Gaussian Puff*. Este modelo está baseado nunha ecuación de Difusión-Advección que depende do tempo, posto que acepta considerar que as emisións desde o punto fonte poden sucederse nun intervalo de tempo, de forma intermitente ou de forma variable no tempo.

A segunda alternativa de estudo pode ser propor o problema en tres dimensións. Aspectos a ter en conta neste caso: o terreo non ten por que ser chan, o dominio non é finito, e as hipóteses para atopar a solución particular non son válidas, por exemplo, en escenarios urbanos. Para esta clase de escenarios, é preciso empregar o modelo baseado na ecuación en derivadas parciais (1.6), o cal supón unha complexidade engadida xa que, en xeral, non dispoñemos de solución explícita. Esta clase de problemas de optimización para os cales temos que resolver unha ecuación en derivadas parciais para avaliar o funcional de coste, enmárcanse dentro da teoría dos problemas de control óptimo (ver por exemplo, a referencia [8]). Que dúbida cabe, que ao non dispor dunha solución exacta do modelo,

necesitamos aproximala dalgunha maneira. Dita aproximación pódese levar a cabo, por exemplo, empregando o método dos elementos finitos (ver, por exemplo, [20]) .

Por último debemos estar abertos a posibilidade de empregar estas técnicas en outros contextos para resolver outro tipo de problemas físicos. Un exemplo de este tipo podería ser o proceso de construír unha ruta de escape en caso de incendios.

Anexo

Anexo A

Método do Gradiente Conxugado con Proxección Ortogonal

Función de cálculo da concentración, $\mathrm{C}(\mathrm{x},\mathrm{y})$

```
% ****************************************************************
% Autora: Eva Miranda Barreiro
% Nome da funcion: conc
% Descripcion: calculo da concentracion asociada a solucion do
% modelo Gaussian Plume para z=Href
% ****************************************************************

% z=Href, sendo Href a altura minima exisida para os bombeiros
% tanto en mulleres como en homes.

function C=conc(x,y)
% Declaracion de variables
global Q K u H Href
C=(Q/(4*pi*K*x))*exp((-u*y^2)/(4*K*x))*(exp((-u*(Href-H)^2)/(4*K*x))+...
    exp((-u*(Href+H)^2)/(4*K*x)));
end
```

Función de cálculo do gradiente da función concentración, $\nabla\mathrm{C}(\mathrm{x},\mathrm{y})$

```
% ****************************************************************
% Autora: Eva Miranda Barreiro
% Nome da funcion: gradconc
% Descripcion: calculo do gradiente da concentracion
%****************************************************************

% z=Href
```

```
8
9  function Gc=gradconc(x,y)
10 % Declaramos as variables
11 global Q H u K Href
12 % Construimos o vector gradiente
13 Gc=zeros(1,2);
14 % Primeira componente
15 Gc(1)=(Q*exp(-(u*y^2)/(4*K*x))*((u*exp(-(u*(H - Href)^2)/(4*K*x))*...
16       (H - Href)^2)/(4*K*x^2) + (u*exp(-(u*(H + Href)^2)/(4*K*x))*...
17       (H + Href)^2)/(4*K*x^2)))/(4*K*x*pi) - (Q*exp(-(u*y^2)/(4*K*x))*...
18       (exp(-(u*(H + Href)^2)/(4*K*x)) + exp(-(u*(H - Href)^2)/(4*K*x))...
19       ))/(4*K*x^2*pi) + (Q*u*y^2*exp(-(u*y^2)/(4*K*x))*...
20       (exp(-(u*(H + Href)^2)/(4*K*x)) + exp(-(u*(H - Href)^2)/(4*K*x)))...
21       )/(16*K^2*x^3*pi);
22 % Segunda componente
23 Gc(2)=-(Q*u*y*exp(-(u*y^2)/(4*K*x))*(exp(-(u*(H + Href)^2)/(4*K*x)) +...
24       exp(-(u*(H - Href)^2)/(4*K*x))))/(8*K^2*x^2*pi);
25 end
```

Representación gráfica da concentración sobre o dominio $\Omega = [0, L_1] \times \left[\frac{-L_2}{2}, \frac{L_2}{2}\right]$

```
1  %*****************************************************************
2  % Autor: Eva Miranda Barreiro
3  % Nombre del script: pinta
4  % Descripcion: Realiza unha grafica da dispersion do contaminante
5  %*****************************************************************
6
7  clear all;
8  format long e;
9
10 % Declaramos as variables globais
11 global K % Coeficiente de difusion
12 global Q % Termino fonte
13 global u % Velocidade do vento
14 global H % Altura do punto de descarga
15 global Href % Altura de referencia para o calculo do coste
16
17 % Asignamos valores
18 K=1.e1;
19 Q=1.e4;
20 u=10;
21 H=2;
22 Href=1.62;
23
24 % Definimos os valores para a representacion
```

```
N=20; %Numero de divisions na malla
LMAX=10; %Dimension do dominio

%Xeramos a malla
[xx,yy] = meshgrid(linspace(0.5,LMAX,N),linspace(-LMAX,LMAX,2*N));

%Calculamos a concentracion nos nodos do mallado
cc=(Q./(4*pi*K*xx)).*exp((-u*yy.^2)./(4*K*xx)).*(exp((-u*(Href-H)^2)./(4*K*
    xx))...
    +exp((-u*(Href+H)^2)./(4*K*xx)));

%Realizamos a representacion grafica
[cs,ch] = contourf(xx,yy,cc);
colormap jet
clabel(cs)
xlabel('Posicion eje x (m)')
ylabel('Posicion eje y (m)')
c = colorbar;
c.Label.String = 'Conentracion de agente quimico (kg/m^3)';
grid on
```

Función de cálculo do funcional de coste, f(x, y)

```
%*****************************************************************
%Autora: Eva Miranda Barreiro
%Nome da funcion: funcoste
%Descripcion: funcion de coste a minimizar
%*****************************************************************

function F=funcoste(x,y)

%Declaracion de variables globais
global lambda1 lambda2 n

%Creamos unha matriz nx1, C, de concentracions
%en cada parella (x,y)
C=zeros(n+2,1);
for i=1:n+2
    C(i)=conc(x(i),y(i));
end

%Inicializamos a variable de saida
F=0.0;

%Calculamos a funcion de coste
```

```
for i=1:n+1 % Calculamos a derivada parcial con respecto ao punto x(i+1)
    % Calculamos a distancia entre o punto (x(i),y(i)) e (x(i+1),y(i+1))
    li=sqrt((x(i+1)-x(i))^2+(y(i+1)-y(i))^2);
    % Calculamos a aportacion da concentracion sobre a curva
    F=F+lambda1*(1/2)*(C(i)+C(i+1))*li;
    % Calculamos a aportacion da distancia total da curva
    F=F+lambda2*li;
end

end
```

Función de cálculo do gradiente da función de coste, $\nabla f(x, y)$

```
%*****************************************************************
%Autora: Eva Miranda Barreiro
%Nome da funcion: gradfuncoste
%Descripcion: gradiente da funcion de coste
%*****************************************************************

function Gf=gradfuncoste(x,y)

%Declaracion de variables globais
global lambda1 lambda2 n

%Creamos o vector columna gradiente
Gf=zeros(2*n,1);

%Creamos unha matriz nx2, GC, que contena o gradiente de concentracin en
%cada parella (x,y)
GC=zeros(n,2);
for i=1:n
    GC(i,:)=gradconc(x(i+1),y(i+1));
end

%Creamos unha matriz nX1, C, que contena a concentracion en cada parella
%(x,y)
C=zeros(n+2,1);
for i=1:n+2
    C(i)=conc(x(i),y(i));
end

%Engadimos as componentes a Gf
for i=1:n %Calculamos a derivada parcial con respecto ao punto x(i+1)
    %Calculamos a distancia entre o punto (x(i),y(i)) e (x(i+1),y(i+1))
    li=sqrt((x(i+1)-x(i))^2+(y(i+1)-y(i))^2);
```

```
33      % Calculamos a distancia entre o punto (x(i+1),y(i+1)) e
34      % (x(i+2),y(i+2))
35      lip1=sqrt((x(i+2)-x(i+1))^2+(y(i+2)-y(i+1))^2);
36      % Calculamos a derivada da distancia entre o punto (x(i),y(i)) e
37      % o punto (x(i+1),y(i+1)), con respecto x(i+1)
38      lixip1=(x(i+1)-x(i))/li;
39      % Calculamos a derivada da distancia entre o punto (x(i),y(i)) e
40      % o punto (x(i+1),y(i+1)), con respecto y(i+1)
41      liyip1=(y(i+1)-y(i))/li;
42      % Calculamos a derivada da distancia entre o punto (x(i+1),y(i+1))
43      % e o punto (x(i+2),y(i+2)), con respecto a x(i+1)
44      lip1xip1=-(x(i+2)-x(i+1))/lip1;
45      % Calculamos a derivada da distancia entre o punto (x(i+1),y(i+1))
46      % e o punto (x(i+2),y(i+2)), con respecto a y(i+1)
47      lip1yip1=-(y(i+2)-y(i+1))/lip1;
48
49      % Calculamos a componente i do gradiente
50      Gf(i)=lambda1*(1/2)*(GC(i,1)*li ...
51                           +GC(i,1)*lip1 ...
52                           +(C(i)+C(i+1))*lixip1 ...
53                           +(C(i+1)+C(i+2))*lip1xip1) ...
54          +lambda2*(lixip1+lip1xip1);
55       % Calculamos a componente n+i do gradiente
56       Gf(i+n)=lambda1*(1/2)*(GC(i,2)*li ...
57                              +GC(i,2)*lip1 ...
58                              +(C(i)+C(i+1))*liyip1 ...
59                              +(C(i+1)+C(i+2))*lip1yip1) ...
60              +lambda2*(liyip1+lip1yip1);
61  end
62  end
```

Test para verificar a definición da función *conc.m*

```
1  %*****************************************************************
2  % Autora: Eva Miranda Barreiro
3  % Descripcion: test para verificar a boa definicion da funcion
4  % concentracion, conc.m
5  %*****************************************************************
6
7  % Cambiamos a precision para os calculos
8  clear all;
9  format long e;
10
11 % Declaramos as variables globais
12 global K % Coeficiente de difusion
```

```
13 global Q % Termino fonte
14 global u % Velocidade do vento
15 global H % Altura do punto de descarga
16 global Href % Altura de referencia para o calculo do coste
17 global lambda1 % Coeficiente asociado a concentracion no coste
18 global lambda2 % Coeficiente asociado a distancia no coste
19 global ndeltac
20
21 % Asignamos valores
22 K=1.e1;
23 Q=1.e4;
24 u=10;
25 H=10;
26 Href=1.62;
27 lambda1=1/2;
28 lambda2=1/2;
29
30 % Xeramos un vector cos incrementos
31 ndeltac=10;
32 deltac=zeros(ndeltac,1);
33 for i=1:ndeltac
34     deltac(i)=(0.1)^i;
35 end
36
37 % Declaramos unha tabla para almacenar os resultados
38 RESULTADOS=zeros(ndeltac,6);
39
40 % Calculamos o gradiente no punto e a concentracion no (5,5)
41 xp=[5,5];
42 gcr11=gradconc(xp(1),xp(2));
43 cr11=conc(xp(1),xp(2));
44 gca11=zeros(size(gcr11));
45
46 % Realizamos un bucle nos delta
47 for i=1:ndeltac
48     % Imprimimos informacion por pantalla
49     disp(['Calculamos la fila ',num2str(i)])
50     % Aproximamos o valor do gradiente empleando diferencias finitas
51     gca11(1)=(conc(xp(1)+deltac(i),xp(2))-cr11)/deltac(i);
52     gca11(2)=(conc(xp(1),xp(2)+deltac(i))-cr11)/deltac(i);
53     % Almacenamos cada fila na matriz RESULTADOS
54     RESULTADOS(i,1)=deltac(i); % delta
55     RESULTADOS(i,2)=gca11(1); % gradiente de concentracion aproximado x
56     RESULTADOS(i,3)=gca11(2); % gradiente de concentracion aproximado y
57     RESULTADOS(i,4)=gcr11(1); % gradiente de concentracion real x
```

```
58      RESULTADOS(i,5)=gcr11(2);   % gradiente de concentracion real y
59      RESULTADOS(i,6)=sqrt((gca11(1)-gcr11(1))*(gca11(1)-gcr11(1))+...
60          (gca11(2)-gcr11(2))*(gca11(2)-gcr11(2)));  % diferencia
61  end
62
63  % Graficamos os erros
64  plot(RESULTADOS(:,1),RESULTADOS(:,6),'*');
```

Test para verificar a definición da función *funcoste.m*

```
1   %*****************************************************************
2   % Autora: Eva Miranda Barreiro
3   % Descripcion: test para verificar a boa definicion do funcional de coste,
4   % funcoste.m
5   %*****************************************************************
6
7   % Cambiamos a precision para os calculos
8   clear all;
9   format long e;
10
11
12  % Declaramos as variables globais
13  global K % Coeficiente de difusion
14  global Q % Termino fonte
15  global u % Velocidade do vento
16  global H % Altura do punto de descarga
17  global Href % Altura de referencia para o calculo do coste
18  global lambda1 % Coeficiente asociado a concentracion no coste
19  global lambda2 % Coeficiente asociado a distancia no coste
20  global n % Numero de nodos
21
22  % Asignamos valores
23  K=1.e1; % Coeficiente de difusion
24  Q=1.e4; % Tasa de descarga
25  u=10; % Velocidad do vento
26  H=2; % Altura a que se produce el vertido
27  Href=1.62; % Altura a que se realiza a medicion (c(x,y,Href))
28  L1=10; % Dimension horizontal do dominio de calculo
29  L2=20; % Dimension vertical do dominio de calculo
30  n=10; % Numero de nodos
31  lambda1=0.1; % Coeficiente asociado ao termino de concentracion
32  lambda2=1-lambda1; % Coeficiente asociado ao termino de distancia
33
34  % Xeramos un vector cos incrementos
35  ndeltac=10;
```

```
deltac=zeros(ndeltac,1);
for i=1:ndeltac
    deltac(i)=(0.1)^i;
end

%Definimos os vectores de posicion x e y
x=10:-1:1;
y=10*ones(size(x));
n=length(x)-2;

%Calculamos o vector gradiente da funcion coste evaluado na ruta
gfcr=gradfuncoste(x,y);
gfca=zeros(size(gfcr));

%Dimensionamos a variable de erros
ERRORES=zeros(ndeltac,2);

%Realizamos un bucle nos delta
for j=1:ndeltac
    %Imprimimos informacion por pantalla
    disp(['Calculamos la fila ',num2str(j)])
    %Acutalizamos a variable de erros
    ERRORES(j,1)=deltac(j);
    ERRORES(j,2)=0.0;
    %Calculamos o gradiente aproximado con diferencias finitas
    for i=1:n %Aproximacion da derivada parcial con respecto a x(i+1)
        %Calculamos o vector modificado para a derivada con respecto a x
        xp=x; xp(i+1)=xp(i+1)+deltac(j);
        yp=y;
        %Calculamos a aproximacion con diferencias finitas
        gfca(i)=(funcoste(xp,yp)-funcoste(x,y))/deltac(j);
        %Calculamos o vecor modificado para a derivada con respecto a y
        xp=x;
        yp=y; yp(i+1)=yp(i+1)+deltac(j);
        %Calculamos a aproximacion con diferencias finitas
        gfca(i+n)=(funcoste(xp,yp)-funcoste(x,y))/deltac(j);
        %Actualizamos a variable de erros
        ERRORES(j,2)=ERRORES(j,2)...
            +(gfca(i)-gfcr(i))^2....
            +(gfca(n+i)-gfcr(n+i))^2;
    end
    %Actualizamos a variable de erros
    ERRORES(j,2)=sqrt(ERRORES(j,2));
end
```

Función de proxección sobre o dominio, P_Ω

```
% ********************************************************
% Autora: Eva Miranda Barreiro
% Nome da funcion: proxeccion
% Descripcion: proxeccion do vector w no dominio obtido no metodo iterativo
% da funcion grad_conx_prox.m
% ********************************************************7

function [xp,yp]=proxeccion(x,y)

    % Declaramos as variables globais
    global L1 % Dimension horizontal do dominio de calculo
    global L2 % Dimension vertical do dominio de calculo

    % Calculamos a proxeccion en funcion dos casos
    if((x<=1)&&(y<=-L2/2)) % Caso A
        xp=1;
        yp=-L2/2;
    elseif((1<x)&&(x<=L1)&&(y<=-L2/2)) % Caso B
        xp=x;
        yp=-L2/2;
    elseif((L1<=x)&&(y<=-L2/2)) % Caso C
        xp=L1;
        yp=-L2/2;
    elseif((L1<=x)&&(-L2/2<=y)&&(y<=L2/2)) % Caso D
        xp=L1;
        yp=y;
    elseif((L1<=x)&&(y>=L2/2)) % Caso E
        xp=L1;
        yp=L2/2;
    elseif((1<=x)&&(x<=L1)&&(y>=L2/2)) % Caso F
        xp=x;
        yp=L2/2;
    elseif((x<=1)&&(y>=L2/2)) % Caso G
        xp=1;
        yp=L2/2;
    elseif((x<=1)&&(-L2/2<=y)&&(y<=L2/2)) % Caso H
        xp=1;
        yp=y;
    else
        xp=x;
        yp=y;
    end
end
```

Función de cálculo para calcular o paso mediante a Regra de Armijo

```
% ******************************************************
% Autora: Eva Miranda Barreiro
% Nome da funcion: armijo
% Descripcion: Algoritmo de Armijo para a busqueda do paso
% ******************************************************

function [alpha_armijo] = armijo(alpha,x,y,gradfc,fc,gamma,delta)
    % Declaramos as variables globais
    global n % Numero de nodos
    global a; % Almacenamos a coordenada x do punto de inicio e fin
    global b; % Almacenamos a coordenada y do punto de unicio e fin

    % Calculamos a norma do gradiente ao cuadrado (en columna)
    d=transpose(gradfc)*gradfc;
    % Inicializamos a variable do bucle
    j=1;
    % Bucle while, salimos cuando se cumpla a condicion de Armijo
    while (j>0)
        % Definimos o novo iterante
        x_new = x-alpha*gradfc(1:n);
        y_new = y-alpha*gradfc(n+1:2*n);

        % Realizamos a proxeccion ortogonal de w
        for i=1:n
          [x_new(i),y_new(i)]=proyeccion(x_new(i),y_new(i));
        end

        % Comprobamos a condicion de Armijo
        if (funcoste([a(1);x_new;b(1)],[a(2);y_new;b(2)])<=fc-gamma*alpha*d)
            j = 0;
            alpha_armijo = alpha;
        else
            alpha = alpha*delta;
        end
    end
end
```

Método do Gradiente Conxugado con Proxección Ortogonal

```
% ******************************************************
% Autora: Eva Miranda Barreiro
% Nome da funcion: metodo
% Descripcion: metodo do gradiente conxugado con proxeccion sobre a funcion
```

```
% funcoste
% ********************************************************
function [ejex,ejey] = metodo(x,y)

    % Declaramos as variables globais
    global n % Numero de nodos
    global a; % Componente x do punto de partida e fin
    global b; % Componente y do punto de partida e fin
    global nitmax % Numero de iteracions maxima para o algoritmo
    global tol  % Tolerancia para o algoritmo iterativo

alpha0=1; % Paso incial para o algoritmo de armijo
delta=0.5; % Constante delta para o algoritmo de Armijo
gamma=1e-2; % Costante gamma para o algoritmo de Armijo
d=0.001; % Paso fixo para o metodo

    %************************
    % Seleccionamos que tipo de paso queremos utilizar
    ipaso=2; % (1.- Paso fijo, 2.- Armijo)

    % Incializamos o metodo
    for k=1:nitmax

        % Evaluamos o funcional de coste
        fc=funcoste([a(1);x;b(1)],[a(2);y;b(2)]);

        % Evaluamos o gradiente do funcional de coste
        gradfc=gradfuncoste([a(1);x;b(1)],[a(2);y;b(2)]);

        % Calculamos la direccion de descenso
        if (ipaso==1)
            desc=-d.*gradfc;

        else
            % Chamamos a Armijo
            alfa=armijo(alpha0,x,y,gradfc,fc,gamma,delta);

            % Actualizamos a direcion de descenso
            desc=-alfa.*gradfc;
        end

        % Calculamos o novo iterante
        w0=[x;y];
        w=w0+desc;

```

```
50          %Realizamos a proxeccion ortogonal de w
51          wp=zeros(size(w0));
52          for i=1:n
53              x1=w(i);
54              y1=w(n+i);
55              [xp,yp]=proyeccion(x1,y1);
56              wp(i)=xp;
57              wp(n+i)=yp;
58          end
59
60          %Test de parada 1 sobre o gradiente
61          temp1=norm(gradfuncoste([a(1);wp(1:n);b(1)],[a(2);wp(n+1:2*n);b(2)]))
        ;
62
63          %Test de parada 2 sobre a diferencia de iterantes
64          temp2=norm(wp-[x;y]);
65
66          %Test de parada 3 sobre a diferencia de costes
67          temp3=abs(funcoste([a(1);wp(1:n);b(1)],[a(2);wp(n+1:2*n);b(2)])-fc);
68
69          %Mostramos a evolucion do algoritmo por pantalla
70          disp('************************************************');
71          disp(['Iteracion ',num2str(k)]);
72          disp(['Coste asociado ',num2str(fc)]);
73          disp(['alpha ',num2str(alfa)]);
74          disp(['Condicion de parada (gradiente) ',num2str(temp1)]);
75          disp(['Condicion de parada (iterantes consecutivos) ',num2str(temp2)])
        ;
76          disp(['Condicion de parada (funcion de coste) ',num2str(temp3)]);
77          disp('************************************************');
78
79          %Comprobamos a condicion de parada
80          if ((temp1<tol)||(temp2<tol)||(temp3<tol))
81  %           x=wp(1:n);
82  %           y=wp(n+1:2*n);
83              break
84          else
85              x=wp(1:n);
86              y=wp(n+1:2*n);
87              alpha0=alfa;
88          end
89      end
90
91      %Final da funcion, asignamos valores as variables de saida
92      ejex=[a(1);x;b(1)];
```

```
93      ejey=[a(2);y;b(2)];
94
95  end
```

Programa principal para a resolución do problema de minimización mediante o Método do Gradiente Conxugado con Proxección Ortogonal

```
1  %*************************************************************
2  %Programa PRINCIPAL para a estimacion da RUTA OPTIMA
3  %Autora: Eva Miranda Barreiro
4  %*************************************************************
5
6  clear all;
7  format long e;
8
9  %Declaramos as variables globais
10 global K %Coeficiente de difusion
11 global Q %Termino fonte
12 global u %Velocidade do vento
13 global H %Altura do punto de descarga
14 global Href %Altura de referencia para o calculo do coste
15 global lambda1 %Coeficiente asociado a concentracion no coste
16 global lambda2 %Coeficiente asociado a distancia no coste
17 global n %Numero de nodos
18 global L1 %Dimension horizontal do dominio de c?lculo
19 global L2 %Dimension vertical do dominio de calculo
20 global nitmax %Numero de iteracions maxima para o algoritmo
21 global tol  %Tolerancia para o algoritmo iterativo
22 global a; %Almacenamos la coordenada x del punto de inicio y fin
23 global b; %Almacenamos la coordenada y del punto de unicio y fin
24
25
26 %Asignamos valores
27 K=1.e1;
28 Q=1.e4;
29 u=20;
30 H=2;
31 Href=1.62;
32 L1=10;
33 L2=20;
34 n=14;
35 lambda1=1;
36 lambda2=1-lambda1;
37 nitmax=100000;
```

```
38 tol=1.e-4;
39 a=zeros(2,1); % Punto de saida
40 b=zeros(2,1); % Punto de chegada
41
42
43 % Asignamos valores o punto de chegada
44 b(1)=1   ;
45 b(2)=0;
46
47 % Inicializamos os puntos da ruta de partida
48
49  % Test basico considerando lambda1=0 e partindo
50  % do camino mais curto desde (10,-10)
51     a(1)=L1;
52     a(2)=-L2/2;
53     x0temp=linspace(a(1),b(1),n+2);
54     y0temp=linspace(a(2),b(2),n+2);
55     x0=transpose(x0temp(2:n+1));
56     y0=transpose(y0temp(2:n+1));
57
58  % Test basico considerando lambda1=0 e partindo
59  % do camino mais longo desde (10,-10)
60 %      a(1)=L1;
61 %      a(2)=-L2/2;
62 %      x0temp=linspace(a(1),b(1),n/2+1);
63 %      y0temp=a(2)*ones(size(x0temp));
64 %      x0temp(n/2+2:n+2)=b(1);
65 %      y0temp(n/2+2:n+2)=linspace(a(2)+1,b(2),n/2+1);
66 %      x0=transpose(x0temp(2:n+1));
67 %      y0=transpose(y0temp(2:n+1));
68
69  % Test basico considerando lambda1=0 e partindo
70  % do camino mais longo desde (10,10)
71 %      a(1)=L1;
72 %      a(2)=L2/2;
73 %      x0temp=linspace(a(1),b(1),n/2+1);
74 %      y0temp=a(2)*ones(size(x0temp));
75 %      x0temp(n/2+2:n+2)=b(1);
76 %      y0temp(n/2+2:n+2)=linspace(a(2)-1,b(2),n/2+1);
77 %      x0=transpose(x0temp(2:n+1));
78 %      y0=transpose(y0temp(2:n+1));
79
80 % Test basico considerando lambda1=lambda2=1/2
81 % partindo do camino mais longo desde (0,10)
82 %      a(1)=10;a(2)=0;
```

```
%       x0temp=linspace(a(1),b(1),n+2);
%       y0temp=zeros(size(x0temp));
%       x0=transpose(x0temp(2:n+1));
%       y0=transpose(y0temp(2:n+1));

% Debuxamos a ruta inicial
plot([a(1);x0;b(1)],[a(2);y0;b(2)])
axis([0,11,-11,11])
%pause;

%Chamamos ao algoritmo de optimizacion
tic
[ejex,ejey]=metodo(x0,y0);
toc

%Graficamos a ruta optima
plot(ejex,ejey)
axis([0,11,-11,11]) %Reaxuste ds eixos na grafica
```

Anexo B

Función *fmincon*

Función obxectivo asociada a *fmincon*

```
%*****************************************************************
%Autora: Eva Miranda Barreiro
%Nome da funcion: objetivo
%Descripcion: funcion que evalua o coste e o gradiente
%*****************************************************************

function [fc,gc]=obxectivo(wp)

    %Declaramos as variables globais
    global n %Numero de nodos
    global a; %Componente x do punto de partida e fin
    global b; %Componente y do punto de partida e fin

    %Chamamos a funcion de coste
    fc=funcoste([a(1);wp(1:n);b(1)],[a(2);wp(n+1:2*n);b(2)]);

    %Comprobamos se e necesario evaluar o gradiente do coste
    if nargout > 1
        %Calculamos o gradiente do coste
        gc=gradfuncoste([a(1);wp(1:n);b(1)],[a(2);wp(n+1:2*n);b(2)]);
    end
end
```

Programa para a resolución do problema de minimización mediante a función *fmincon*

```
%*****************************************************************
%Autora: Eva Miranda Barreiro
%Descripcion: Programa no que resolvemos o problema de minimizacion a
```

```
% traves da funcion, fmincon.
%*****************************************************************

clear all;
format long e;

%Chamamos ao arquivo de datos
datos

%Asignamos valores o punto de partida e a proxeccion do punto de
%vertido (imonos  mover polo plano z=Href)
a=zeros(2,1);
b=zeros(2,1);
b(1)=1;
b(2)=0;

%Inicializamos os puntos do camino de partida

    %Test basico considerando lambda1=0 e partindo
    %do camino mais curto desde (10,-10)
%     a(1)=L1;
%     a(2)=-L2/2;
%     x0temp=linspace(a(1),b(1),n+2);
%     y0temp=linspace(a(2),b(2),n+2);
%     x0=transpose(x0temp(2:n+1));
%     y0=transpose(y0temp(2:n+1));

    %Test basico considerando lambda1=0 e partindo
    %do camino mais longo desde (10,-10)
%     a(1)=L1;
%     a(2)=-L2/2;
%     x0temp=linspace(a(1),b(1),n/2+1);
%     y0temp=a(2)*ones(size(x0temp));
%     x0temp(n/2+2:n+2)=b(1);
%     y0temp(n/2+2:n+2)=linspace(a(2)+1,b(2),n/2+1);
%     x0=transpose(x0temp(2:n+1));
%     y0=transpose(y0temp(2:n+1));

     %Test basico considerando lambda1=0 e partindo
    %do camino mais longo desde (10,10)
%     a(1)=L1;
%     a(2)=L2/2;
%     x0temp=linspace(a(1),b(1),n/2+1);
%     y0temp=a(2)*ones(size(x0temp));
%     x0temp(n/2+2:n+2)=b(1);
```

```
%       y0temp(n/2+2:n+2)=linspace(a(2)-1,b(2),n/2+1);
%       x0=transpose(x0temp(2:n+1));
%       y0=transpose(y0temp(2:n+1));

      % Test basico considerando lambda1=lambda2=1/2
      % partindo do camino mais curto desde (0,10)
      a(1)=10;a(2)=0;
      x0temp=linspace(a(1),b(1),n+2);
      y0temp=zeros(size(x0temp));
      x0=transpose(x0temp(2:n+1));
      y0=transpose(y0temp(2:n+1));
% Inicializamos o vector de partida
w0=[x0;y0];


% Inicializamos as opciones para o algoritmo iterativo
Aneq = []; % Matriz asociada as restriccions de desigualdade
bneq = []; % Segundo membro asociado as restricciones lineais
Aeq = []; % Matriz asociada as restriccions de igualdade
beq = []; % Segundo miembro asociado as restriccions de igualdade
nonlcon = []; % Restricciones non lineais de desigualdade
lb=zeros(size(w0)); % Inicializamos as restriccions da cota inferior
lb(1:n)=1.0; % Cota inferior para a componente x
lb(n+1:2*n)=-L2/2; % Cota inferior para a componente y
ub=zeros(size(w0)); % Inicializamos as restriccions da cota superior
ub(1:n)=L1; % Cota superior para a componente x
ub(n+1:2*n)=+L2/2; % Cota superior para a componente y
fun = @obxectivo; % Funcion que evalua o coste e o gradiente

% Establecemos as opcions do algoritmo de optimizacion
 options = optimoptions('fmincon','Display','iter',...
     'SpecifyObjectiveGradient',true,'Algorithm','interior-point','
    MaxIterations',2000,'MaxFunctionEvaluations',10000);
% options = optimoptions('fmincon','Display','iter',...
%     'SpecifyObjectiveGradient',true,'Algorithm','trust-region-reflective','
    MaxIterations',1000);
% options = optimoptions('fmincon','Display','iter',...
%    'SpecifyObjectiveGradient',true,'Algorithm','sqp');
% options = optimoptions('fmincon','Display','iter',...
%     'SpecifyObjectiveGradient',true,'Algorithm','active-set');

% Dimensionamos as variables asociadas ao experimento
%m=10; % Numero de componentes do vector de lambda1
% wop=zeros(2*n,m); % Matriz cos caminos optimos
% lambda=linspace(0.0,1.0,m);
```

```
92 %wop=zeros(2*n,1);
93 %for i=1:m
94 %      %Modificamos os valores de lambda 1 e 2
95 %      lambda1=lambda(i);
96 %      lambda2=1-lambda1;
97       %Resolvemos o problema de optimizacion
98 %      wop(:,i)=fmincon(fun,w0,Aneq,bneq,Aeq,beq,lb,ub,nonlcon,options);
99 %end
100 tic
101 wop=fmincon(fun,w0,Aneq,bneq,Aeq,beq,lb,ub,nonlcon,options);
102 toc
103 %Graficamos os resultados
104 %for i=1:m
105 %      pinta(wop(:,i))
106 %      pause
107 %end
108 pinta(wop)
```

Bibliografía

[1] Boundary-layer diffusion modelling: The gaussian plume approach versus the spectral solution. *Boundary-Layer Meteorology*, 12(2), 1977.

[2] Ntp 334: Planes de emergencia interior en la industria química. Technical report, Instituto Nacional de Seguridad e Higiene en el trabajo, 1991.

[3] A model for simulating atmospheric dispersion in low-wind conditions. *International Journal of Environment and Pollution*, pages 69–79, 2001.

[4] Montserrat Aguirre. La ciudad de méxico registra una inversión térmica que puede durar todo el sábado. https://aristeguinoticias.com/0206/mexico/la-ciudad-de-mexico-registra-una-inversion-termica-que-puede-durar-todo-el-sabado/, 2018.

[5] Dimitri P. Bertsekas. *Nonlinear programming.* Athena Scientific, 1999.

[6] G. T. Csanady. Crosswind shear effects on atmospheric diffusion. *Atmospheric Environment*, pages 221–232, 1972.

[7] Richard Haberman. *Applied partial differential equations: with Fourier series and boundary value problems.* Pearson Prentice Hall, 4 edition, 2004.

[8] J.-L. Lions. *Optimal control of systems governed by partial differential equations.* Translated from the French by S. K. Mitter. Die Grundlehren der mathematischen Wissenschaften, Band 170. Springer-Verlag, New York-Berlin, 1971.

[9] J.Ñocedal and S. Wright. *Numerical Optimization.* Springer series in operations research. Springer, 2 edition, 2006.

[10] Mattheij R., Rienstra S.W., and Boonkkamp J.H.M. *Partial Differential Equations: Modeling, Analysis, Computation.* SIAM Monographs on Mathematical Modeling and Computation, 2005.

[11] O.F.T. Roberts. The theoretical scattering of smoke in a turbulent atmosphere. *Proceedings of the Royal Society A: Mathematical*, pages 640–654, 1923.

[12] Joel L. Schiff. *The Laplace transform.* Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1999. Theory and applications.

[13] Laurent Schwartz. *Théorie des distributions.* Publications de l'Institut de Mathématique de l'Université de Strasbourg, No. IX-X. Nouvelle édition, entiérement corrigée, refondue et augmentée. Hermann, Paris, 1966.

[14] John H. Seinfeld and N. Pandis Spyros. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change.* Wiley-Interscience, 1997.

[15] Ivar Stakgold. *Boundary Value Problems of Mathematical Physics*, volume 2 of *Classics in Applied Mathematics.* Society for Industrial Mathematics, 1987.

[16] J. Stockie. The mathematics of atmospheric dispersion modeling. *SIAM Review*, 53(2):349–372, 2011.

[17] O. G. Sutton. A theory of eddy diffusion in the atmosphere. *The Royal Society*, 135(826):143–165, 1932.

[18] Jose Varela. Efectos de la contaminación atmosférica. el primer episodio de smog fotoquímico de la historia; los angeles, 1943. https://ahombrosdegigantescienciaytecnologia.wordpress.com/2015/07/26/contaminacion-atmosferica-el-primer-episodio-de-smog-fotoquimico-de-la-historia-los-angeles-1943, 2015.

[19] J. M. Viaño and M. Burguera. *Lecciones de métodos numéricos 3, Interpolación.* Tórculo edicións, first edition, 1999.

[20] O. C. Zienkiewicz, R. L. Taylor, and P.Ñithiarasu. *The finite element method for fluid dynamics.* Elsevier/Butterworth Heinemann, Amsterdam, seventh edition, 2014.