Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un algoritmo que integra una red de sensores para monitorización de amenazas NRBQ tipo químico

Grado en Ingeniería Mecánica

ALUMNO: Juan Manuel Ortega Muñiz

DIRECTORES: Javier Martínez Torres

Francisco Javier Fernández Fernández

CURSO ACADÉMICO: 2017-2018

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de un algoritmo que integra una red de sensores para monitorización de amenazas NRBQ tipo químico

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General / Infantería de Marina

Universidade Vigo

RESUMEN

El modelo gaussiano de dispersión de contaminantes permite estudiar el transporte de agentes químicos en la atmósfera debido a la dispersión turbulenta y advección del viento. Este modelo puede ser usado para determinar la fuente de emisión, empleando para ello mediciones obtenidas por sensores instalados en diferentes puntos. Esto resulta de especial interés en la defensa NRBQ para controlar y evaluar el riesgo de una amenaza de este tipo como resultado de un ataque químico. Precisamente, es un problema ya expuesto en la línea 5.2 de la Estrategia de Tecnología e Innovación para la Defensa (ETID 2015) en la que se propone como reto tecnológico la integración de detección NRBQ en redes de sensores mediante algoritmos para vigilar y predecir la evolución de esta clase de ataques. En este trabajo se realizará un análisis matemático del problema con el objetivo de determinar la fuente de emisión, es decir, la ubicación y la cantidad del agente químico vertido. A continuación, se desarrollará un algoritmo que integre las medidas obtenidas por una red de sensores como herramienta de prevención y alerta ante un ataque de tipo químico.

PALABRAS CLAVE

Agentes químicos, software NRBQ, modelo de dispersión, MATLAB, mínimos cuadrados, redes de sensores.

AGRADECIMIENTOS

A mi tutor Fran, ambos comenzamos un nuevo ciclo de nuestra vida después de varios años en esta Escuela. Ha conseguido que este trabajo no haya sido un objetivo, sino un camino agradable. Agradezco su completa ayuda y le deseo lo mejor en su nueva etapa.

A mi tutor Javier, su asignación como nuevo tutor durante las últimas semanas del trabajo aportaron ideas nuevas que sin duda han mejorado el resultado.

A Riveiro, le agradezco su atención y apoyo en los primeros años de mi formación.

A JJ, sin sus consejos quizás no estaría escribiendo estas líneas. Él sabe que en el jardín de un universo cuántico, los senderos se bifurcan constantemente.

A JM, el ejemplo a seguir más cercano y que admiro.

A Raquel, por demostrarme siempre que es una amiga de verdad.

Finalmente, a mi familia, porque sé que siempre estarán a mi lado.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contextualización

Los agentes químicos de guerra se definen como cualquier sustancia con propiedades tóxicas capaz de matar o dañar a un enemigo empleados en escenarios de guerra. La proliferación en masa de este tipo de armas comienza en la Primera Guerra Mundial [1]. Desde entonces, se han empleado tanto en escenarios de guerra como en actos terroristas. El empleo de este tipo de armas siempre se ha asociado a afectos devastadores en el campo de batalla, alcanzado una crueldad innecesaria. Por ello, los esfuerzos internacionales para su prohibición han estado presentes en acuerdos sobre armamento, culminando con la firma de la Convención sobre las Armas Químicas (CAQ), contemplando la prohibición del desarrollo, producción, almacenamiento y uso de los agentes químicos de guerra y su destrucción.

Sin embargo, la fabricación de algunos de estos agentes químicos no se puede prohibir debido a su importante uso en la industria. Estos químicos siguen siendo una amenaza, especialmente de países que no cuentan con tecnología nuclear, dado que son fáciles de producir, de bajo coste económico y producen efectos devastadores. Además, pueden ser usados eficazmente como armas de ataques terroristas de pequeña escala [2].

1.2 Motivación y objetivos

La Organización para la Prohibición de las Armas Químicas (OPAQ) es una organización internacional cuya principal función es asegurar la aplicación de los objetivos de la Convención sobre las Armas Químicas. Su misión es la de eliminar todo tipo de armas químicas en todo el mundo, comprueba que se destruyan todas las armas químicas existentes y que no se produzcan. Dicha organización propone a los estados miembros el desarrollo de medios de protección frentes a agentes químicos, por ejemplo, mejores sistemas de detección de este tipo de amenazas [3].

Figura 1-1 Logo de la Organización para la Prohibición de las Armas Químicas (OPCW, por sus siglas en inglés) [3].

En el ámbito nacional también encontramos una demanda de sistemas de detección o alerta temprana de agentes NRBQ. La Estrategia de Tecnología e Innovación para la Defensa (ETID-2015), propone un marco general en el que deben moverse los distintos planes dedicados a la I+D+i de defensa, para conseguir una evolución de las capacidades militares de las Fuerzas Armadas [4]. En su apartado 5.2.4, propone como reto tecnológico la integración de detección de amenazas NRBQ en una red de sensores,

implementando el desarrollo de algoritmos en un software apropiado que permita predecir la evolución del riesgo mediante herramientas de modelado y simulación.

Figura 1-2 Estrategia de Tecnología e Innovación para l Defensa (ETID-2015) [5].

Estas necesidades establecidas tanto a nivel internacional como a nivel nacional, resultan el principal objetivo por el que desarrollamos este proyecto. Tomando como base la publicación [6], se analizará si el modelo matemático que se emplea para simular la evolución del área del peligro de un agente químico no persistente, *Gaussian Puff*, es válido para obtener la fuente de emisión de dicho agente. Es decir, se busca el problema inverso, mediante un algoritmo implementado en una red de sensores que proporcione medidas de la concentración de la amenaza química.

La alerta temprana de un ataque químico supone una ventaja crucial para llevar a cabo las medidas de defensa oportunas y para monitorizar la evolución de la zona de peligro, estableciendo así una zona segura.

El sistema desarrollado deberá obtener las coordenadas y la cantidad de la fuente de emisión para un escenario definido por una red fija de sensores instalados previamente. Para cumplir este objetivo principal, deberemos cumplir los siguientes:

- Analizar el modelo de predicción del área de peligro mediante el teorema de la función inversa.
- Comprobar el comportamiento del programa desarrollado en escenarios académicos para validarlo y extender posteriormente su uso a escenarios realistas.
- Analizar el área que los sensores pueden abarcar para obtener una solución válida.

1.3 Descripción de la memoria

Conocida ya la motivación de este trabajo y marcados los objetivos que se esperan cumplir, se describen, a continuación, los diferentes apartados que componen la memoria, para que el lector pueda tener una visión global del trabajo desarrollado.

- En el primer capítulo se presenta una breve contextualización sobre el empleo de armas químicas y su actual prohibición regulada a nivel internacional. Además, se comenta la motivación y objetivos del problema.

- En el segundo capítulo se trata sobre el estado del arte. Se comentarán hechos puntuales de la historia de los agentes químicos de guerra, su clasificación y sus principales características.
- En el tercer capítulo se refleja el desarrollo del problema. Comenzamos analizando el modelo y aplicando el teorema de la función inversa para determinar si nuestro problema tiene solución, desde un plano puramente teórico. A continuación, se explican los métodos y *scripts* llevados a cabo para obtener dicha solución mediante el programa MATLAB.
- En el cuarto capítulo se describen las pruebas realizadas, simulando un escenario con medidas sintéticas. Para estimar la capacidad del algoritmo de obtener soluciones válidas, en función del número de sensores y de medidas, se realizará un mallado del escenario, comprobando el error en cada uno de los puntos en los que obtenemos una solución respecto a la fuente de emisión sintética.
- En el quinto capítulo se discutirá el alcance de los objetivos establecidos inicialmente, así como las limitaciones encontradas durante el desarrollo del trabajo. También se definirán las líneas de futuro que hayan surgido del proyecto y sus posibles mejoras.
- Finalmente, existen un apartado de anexos en el que se incluyen todos los *scripts* desarrollados.

2 ESTADO DEL ARTE

2.1 Introducción

Los agentes químicos de guerra se han empleado desde tiempos inmemorables. Los primeros escritos que hacen referencia a este tipo de ataques se encuentran en la literatura China, seguidos por la cultura griega y la romana. Durante el siglo XIX los rápidos avances de la química y su industria trajeron como consecuencia graves accidentes relacionados con químicos letales. Aumenta así el conocimiento de sus efectos toxicológicos y la posibilidad de su producción a gran escala. El primer uso de químicos como armas de destrucción masiva se remonta a la Primera Guerra Mundial por parte de las fuerzas militares alemanas en Ypres, Bélgica (1915). Comienza así un interés militar en desarrollar armas químicas cada vez más eficientes y letales, así como medios de difusión apropiados para su empleo en conflictos armados [2].

Figura 2-1 Infantería australiana empleando máscaras de gas en Ypres, Bélgica, septiembre 1917 [7].

Durante la Segunda Guerra Mundial no se emplearon agentes químicos en el campo de batalla por miedo de que el enemigo contara con otros químicos aún más letales. Después de la Segunda Guerra Mundial, continúa su uso tanto en guerras como en ataques terroristas, como los llevados a cabo por la secta *Aum Shinrikyo* en Matsumoto y Tokio empleando gas Sarín [1].

Estos hechos propician un esfuerzo a nivel mundial por prohibir el uso de los agentes químicos de guerra. En 1993 se firma la Convención sobre Armas Químicas, un tratado internacional de control de armamento que ilegaliza la producción, almacenamiento y uso de armas químicas [3]. Entró en vigor el 29 de abril de 1997, estableciendo que todos los países miembros debían destruir sus reservas de agentes químicos de guerra en un periodo de diez años.

2.2 Clasificación y descripción de los agentes químicos de guerra

La Organización del Tratado del Atlántico Norte u OTAN (NATO, por sus siglas en inglés, *North Atlantic Treaty Organization*) clasifica los agentes químicos de guerra como agentes vesicantes, agentes nerviosos, agentes pulmonares, asfixiantes y agentes incapacitantes o que alteran el comportamiento [2].

2.2.1 Agentes vesicantes

Los agentes vesicantes pertenecen a un conjunto de químicos que causan graves ampollas al entrar en contacto con la piel. Aunque no sean tan letales como para causar la muerte del sujeto afectado, cumplen el objetivo de incapacitar al enemigo y sobrecargar los servicios de atención médica. A continuación, mostramos los tipos de agentes vesicantes.

2.2.1.1 Mostaza sulfurada o gas mostaza

La mostaza sulfurada, conocida comúnmente como gas mostaza por su característico olor a mostaza o parecido al del ajo, es un agente alquilante (en química orgánica, transferencia de un grupo alquilo de una molécula a otra) capaz de causar morbidez (que es indicio o causa de enfermedad) a corto y largo plazo. Este agente descubierto accidentalmente en 1822 fue empleado por primera vez por el ejército alemán en 1917 en el campo de batalla. Se le atribuye el 70% del más de millón de muertes relacionadas con gases durante la Primera Guerra Mundial.

El gas mostaza es un líquido aceitoso de color amarillo pálido que se vaporiza a los 25 ºC y se descompone a la temperatura de 217.5 ºC, de ahí que se use la incineración a alta temperatura para su destrucción. Por consiguiente, en ambientes húmedos y fríos es un líquido y es fácil de evaporizar en ambientes secos y cálidos. Es más pesado que el aire, con una densidad 5.6 veces la del aire, 1.27 kg/cm^3. Presenta un olor a mostaza en su forma impura, pero la forma pura de este agente es incolora y sin ningún olor. Se disuelve fácilmente en agua y grasa. En forma gaseosa, penetra con facilidad la ropa común. No tiene ningún uso industrial.

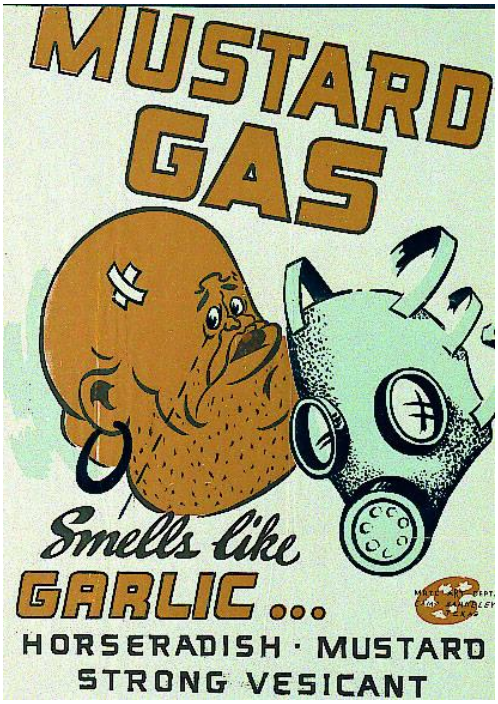

Figura 2-2 Póster de identificación del gas mostaza durante la Segunda Guerra Mundial [8].

Sobre su mecanismo de toxicidad en humanos, el gas mostaza daña todas las zonas epiteliales expuestas. Los efectos aparecen entra las dos y las doce horas tras la dosis de exposición. Aunque el mecanismo exacto de actuación todavía no está determinando propiamente, la mayoría de autores creen

que se disuelve en medios acuosos, como el sudor, formando rápidamente iones reactivos que reaccionan con el ADN que rápidamente dividen las células causando muerte celular y reacciones inflamatorias.

La exposición a este agente produce una alta morbidez y secuelas psicológicas, pero presenta bajo índice de mortalidad.

No existe un antídoto específico. Se aconseja atención de personal cualificado en incidentes NRBQ propiamente equipados que extraigan correctamente la ropa del afectado, seguido de una ducha de agua para ayudar al proceso de descontaminación.

2.2.1.2 Lewisita

La lewisita se clasifica como agente abrasante. Su origen reside en la Primera Guerra Mundial, pero al tiempo que fue sintetizada, la guerra había acabado. Durante la Segunda Guerra Mundial se comprobó que era menos eficiente que el gas mostaza, por lo que no se no se fabricó a gran escala. Además, se hidroliza en ambientes húmedos, lo que se traduce en un bajo rendimiento para escenarios operativos. A pesar de no haber sido empleado durante tiempos de guerra, se clasifica como agente químico de guerra.

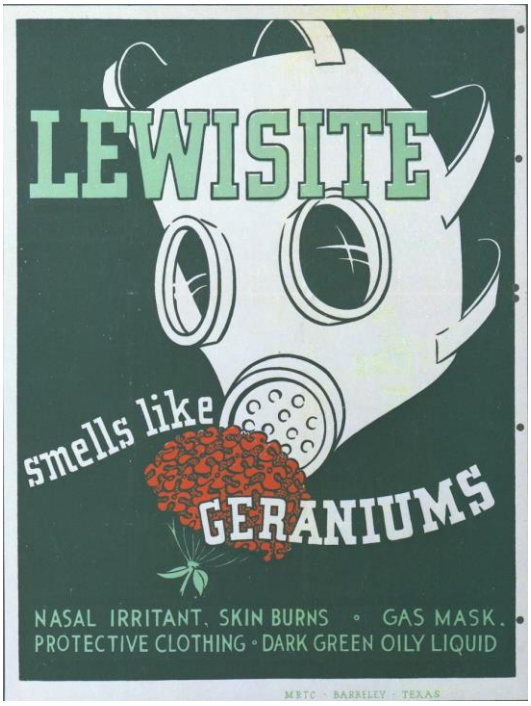

Figura 2-3 Póster de identificación de la lewisita durante la Segunda Guerra Mundial [8].

Su densidad es de 1.89 kg/cm^3. El punto de fusión tiene lugar a los -18 ºC y el de ebullición a los 190 ºC. En agua se hidroliza formando ácido hidroclórico, puede formar arseniato trisódico en disoluciones alcalinas, con propiedades venenosas.

Fue empleado por Estados Unidos como anticongelante para el gas mostaza y para penetrar ropa de protección en situaciones especiales, pero en 1950 se declara como obsoleto. En la actualidad no tiene ningún uso industrial.

Su mecanismo de toxicidad se caracteriza por su facilidad para penetrar la ropa común y materiales como la goma. Sus síntomas son similares al resto de agentes vesicantes. Para su tratamiento también se recomienda la eliminación de la ropa que porte el afectado y un baño de agua abundante como primeras medidas. Además, existe un antídoto de origen británico denominado anti-Lewisita empleado para niveles tóxicos graves.

2.2.1.3 Mostaza nitrogenada

La mostaza nitrogenada es otro agente químico alquilante, análogo al gas mostaza. Existen tres versiones etiquetadas militarmente como HN-1, HN-2 y HN-3. Fueron producidos entre 1920 y 1930 como potenciales agentes químicos de guerra. Durante la Segunda Guerra Mundial, Estados Unidos produjo casi cien toneladas de HN-1, aunque nunca fueron empleados. El HN-1 se desarrolló originalmente como tratamiento para las verrugas, pero acabó clasificado en la categoría de agente químico.

En cuanto a sus propiedades, el HN-1 presenta un débil olor a pescado. Se disuelve fácilmente en agua y se descompone a partir de los 194 ºC. El HN-2 tiene un olor afrutado a altas concentraciones y un olor jabonoso a bajas concentraciones. Por último, el HN-3 no posee olor en su forma pura y es el más estable de las mostazas nitrogenadas. Se descompone a partir de los 256 ºC. Su presión de vaporización es mucho más baja que las del HN-1 y HN-2 y es insoluble en el agua.

No tienen ningún uso industrial y su mecanismo de actuación es similar al del gas mostaza. Los síntomas aparecen después de varias horas después de la exposición al agente. La concentración de la dosis determina lo rápido que puedan aparecer los síntomas. Dichos síntomas también son similares, afectando a la piel, los ojos, el tracto respiratorio y el sistema gastrointestinal. Tampoco existe un tratamiento específico.

2.2.2 Agentes nerviosos

Este tipo de agentes son parecidos a los pesticidas organofosforados, que son compuestos orgánicos degradables que contienen enlaces de fósforo y carbono. Son unos de los tipos de agentes químicos más letales. Se clasifican según la OTAN de la siguiente forma:

- Serie G: GB (Sarín), GD (Somán) y GA (Tabún).
- Serie V: VE, VX, VG y VM.

Se estima que Alemania fabricó unas 12,000 toneladas de estos agentes químicos durante la Segunda Guerra Mundial, aunque no se emplearon en el campo de batalla. Se emplearon por primera vez en la Primera Guerra del Golfo por parte de Iraq contra Irán.

Figura 2-4 Soldados iraquíes empleando máscaras de gas en la Primera Guerra del Golfo [9].

Para su fabricación se siguen procesos químicos similares a los empleados en insecticidas agrícolas. Todos los agentes nerviosos son líquidos a temperatura y presión estándar. Su alta volatilidad los convierte en un arma muy poderosa. Como ejemplo, una dosis de $0.5\,mg$ de GB (Sarín) es letal en humanos. El agente VX es el menos volátil, pudiendo permanecer en tierra durante 24 horas. Además de como agente químico de guerra, los agentes nerviosos se emplean ampliamente como pesticidas para la agricultura.

En cuanto a su mecanismo de actuación, los agentes nerviosos pueden causar una crisis colinérgica, que consiste en una sobre estimulación de la unión entre el músculo y el nervio, debido a un exceso del neurotransmisor llamado acetilcolina. Además, el agente VX también puede causar daño pulmonar agudo.

Los casos agudos de envenenamiento por agentes nerviosos se deben tratar mediante equipos de respiración auxiliares de descontaminación y antídotos específicos.

2.2.1 Agentes asfixiantes

Se clasifican en asfixiantes simples y asfixiantes químicos. Los asfixiantes simples, como son el metano y el nitrógeno, desplazan el oxígeno en aire inspirado, resultando en deficiencia de oxígeno e hipoxemia (disminución anormal del nivel de oxígeno en la sangre). Los asfixiantes químicos, como los cianuros, interfieren con el transporte de oxígeno a nivel celular causando hipoxia (disminución de la difusión de oxígeno en los tejidos y en las células). Los químicos asfixiantes empleados como agentes químicos de guerra son el cloruro de cianógeno, el ácido cianhídrico y el arsano.

2.2.1.1 Cloruro de cianógeno

El cloruro de cianógeno fue empleado por Francia durante la Primera Guerra Mundial. Fue un agente químico muy efectivo por dos razones. La primera es que las máscaras de protección de esa época no eran efectivas contra este agente, lo que implicó su producción en masa por parte de los Estados Unidos. La segunda razón es que no es un agente inflamable, permitiendo su uso simultáneo con cargas explosivas. Después de la Segunda Mundial, su uso fue desplazado por los agentes nerviosos, de actuación más rápida. Este agente interfiere con la respiración aeróbica a nivel celular, resultando en la incapacidad de emplear el oxígeno y la muerte celular. Una concentración suspendida en el aire de $270\ ppm$ resulta suficiente para ser letal.

2.2.1.2 Ácido cianhídrico

El ácido cianhídrico fue empleado en grandes cantidades durante la Primera Guerra Mundial por parte del ejército francés, pero resultó ser menos eficiente comparado con otros agentes químicos de guerra por su tendencia a evaporarse rápidamente. Es altamente tóxico y para una concentración adecuada, causa la muerte rápidamente. Al igual que el cloruro de cianógeno, se emplea en la industria para la síntesis de herbicidas y para la limpieza de metales. Su alta volatilidad lo convierte en un agente químico difícil de emplear en la guerra.

2.2.1.3 Arsano

Respecto al arsano, es una de las formas de arsénico más tóxica. Como agente químico, fue estudiado exhaustivamente durante la Segunda Guerra Mundial, pero su bajo nivel de toxicidad previno su empleo en la guerra. Sin embargo, sigue siendo una amenaza como ataque terrorista de pequeña escala, por lo que es clasificado por la OTAN como agente químico de guerra. La inhalación de este agente provoca la destrucción de glóbulos rojos, causando hipoxia. No existe un antídoto específico.

2.2.2 Tóxicos pulmonares

Los tóxicos pulmonares se clasifican como aquellos que producen una respuesta inflamatoria en los pulmones al ser inhalado. Sus efectos se manifiestan provocando edemas pulmonares y alterando el intercambio de gases. Los principales agentes de este tipo son el cloro, la cloropicrina y el fosgeno.

2.2.2.1 Cloro

Es su estado gaseoso, el cloro es venenoso. Durante la Primera Guerra Mundial Alemania empleó el cloro como agente químico y, aunque no tuvo mucho éxito en la guerra, abrió el camino hacia la producción en gran escala de agentes químicos de guerra tanto por Alemania como por los Aliados. En la actualidad, es un químico de amplio uso comercial.

El cloro es presurizado y enfriado de forma que pueda ser almacenado en estado líquido. Cuando el líquido se libera, se transforma rápidamente en un gas de color verde azulado y con un olor muy molesto. Es conocido por su uso como agente blanqueador en la industria textil y del papel. También se emplea para fabricar pesticidas, goma y disolventes. También es ampliamente empleado de forma doméstica para el mantenimiento de piscinas. Respecto a su tratamiento, no existe ningún antídoto específico.

2.2.2.2 Cloropicrina

El vapor de cloropicrina es altamente venenoso cuando se inhala. Se empleó durante la Primera Guerra Mundial como agente químico de guerra por parte de los británicos, alemanes y franceses. Tras la Segunda Guerra Mundial su uso quedó obsoleto. Es más tóxico que el cloro, pero menos que el fosgeno. Se trata de un líquido aceitoso e incoloro.

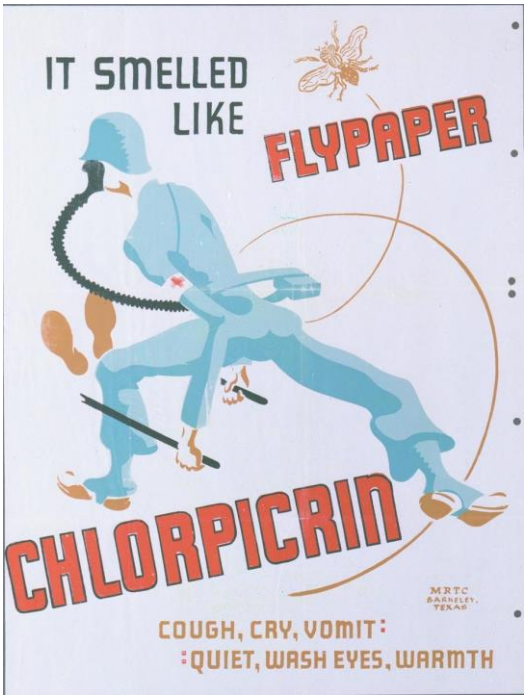

Figura 2-5 Póster de identificación de la cloropicrina durante la Segunda Guerra Mundial [8].

En la industria se emplea para la fabricación de fumigantes, fungicidas e insecticidas, así como para la exterminación de ratas. También se emplea para fumigar y esterilizar tierras de cultivo y semillas.

Debido al pequeño tamaño de sus moléculas, la cloropicrina penetra los filtros de las máscaras de gas, haciendo que la víctima se quite su máscara de gas, por lo que normalmente se mezcla con otro agente químico.

2.2.2.3 Fosgeno

Siendo más letal que el cloro, fue usado por Alemania en 1915 durante la Primera Guerra Mundial, causando 1069 bajas y 120 muertes reportadas. Consecuentemente, los Aliados también hicieron uso de este agente químico, al que se le atribuye el 85% de las bajas de la Primera Guerra Mundial. Aparte de su uso como arma, tiene un amplio uso en la industria. Se almacena en estado líquido a bajas temperaturas con un punto de ebullición de $8.2°C$. Por lo tanto, a temperatura ambiente es un gas venenoso, formando una nube incolora de color blanco o amarillo pálido, con un olor agradable a heno recién sesgado, aunque en grandes concentraciones genera un olor desagradable.

En la industria química se emplea para la fabricación de farmacéuticos, colorantes, pesticidas y poliuretano para gomaespuma. No existe ningún antídoto específico contra este tipo de agente químico.

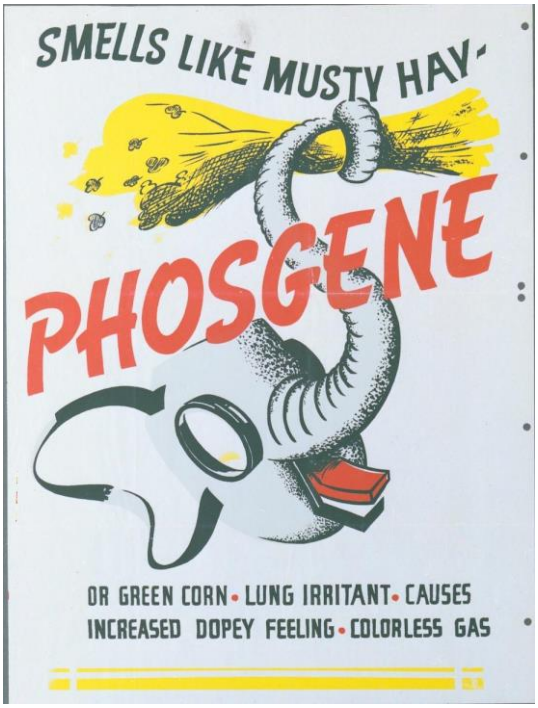

Figura 2-6 Póster de identificación del fosgeno durante la Segunda Guerra Mundial [8].

2.2.3 Agentes que alteran el comportamiento

Después de la Segunda Guerra Mundial, el ejército de los Estados Unidos investigó un amplio rango de posibles químicos no letales que alterasen el comportamiento e incapaciten a la víctima. BZ es el código de clasificación OTAN por el que se conoce al único agente químico de guerra de este tipo.

Se cree que este agente ha sido almacenado en grandes cantidades por Iraq. También se piensa que fue este químico de guerra el empleado en el asalto de las Fuerzas Especiales Rusas para liberar a los rehenes capturados en un teatro de Moscú el 26 de octubre del 2002, dado que la naturaleza exacta de dicho gas permanece desconocida. Se estima que de las 127 muertes, 103 se atribuyen al BZ.

El BZ es un gas inodoro capaz de persistir de tres a cuatro semanas en aire húmedo. Entre sus efectos se incluyen alucinaciones, confusión, agitación, temblores, estupor y coma. Resulta extremadamente persistente en superficies como el suelo o el agua. No tiene otro uso más que como arma química. Tampoco existe ningún antídoto específico contra este agente.

3 DESARROLLO DEL TFG

3.1 Descripción general

Ante un incidente de tipo NRBQ es de vital importancia predecir el área de peligro. La localización de la fuente emisión y su cantidad vertida resulta de gran interés para la alerta temprana ante este tipo de amenazas, acelerando así las medidas de defensa y neutralización oportunas. En la publicación [10] se describe cómo emplear el modelo *Gaussian Puff* para simular la evolución del área de peligro de un ataque de agente químico no persistente. Empleando este modelo, se trata de caracterizar de forma conveniente el incidente mediante una función que mida la diferencia entre las mediciones obtenidas por los sensores y las que predice nuestro modelo.

En la sección 3.2 hablaremos sobre el modelo de dispersión de un contaminante y las hipótesis aplicadas para obtener una solución particular, en la sección 3.3 mostraremos cómo plantear el problema, en la sección 3.4 analizaremos matemáticamente el problema mediante el teorema de la función implícita, así como las funciones potenciales de obtener una solución, y en la sección 3.5 explicaremos un conjunto de *scripts* para resolver el problema.

3.2 Modelo de dispersión de un contaminante

De forma genérica, partiremos de la ecuación de reacción-difusión-convección genérica siguiente [11]:

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\vec{u}) = \nabla \cdot (K\nabla \mathrm{C}) + \mathrm{S}$$

donde:

- $u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$ es la velocidad del viento $\left(\frac{m}{s}\right)$
- $K = \begin{pmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{pmatrix}$ es la matriz de difusividades $(\frac{m^2}{s})$
- C es concentración del contaminante $(\frac{Kg}{m^3})$
- S es el término fuente $(\frac{Kg}{m^3 s})$

Ahora bien, si suponemos que $\nabla \cdot \vec{\mathrm{V}} = \frac{\partial V_x}{\partial \mathrm{x}} + \frac{\partial V_y}{\partial \mathrm{y}} + \frac{\partial V_z}{\partial \mathrm{z}} = 0$, se tendrá que:

$$\nabla \cdot (\mathrm{C}\vec{u}\,) = \frac{\partial C u_x}{\partial \mathrm{x}} + \frac{\partial C u_y}{\partial \mathrm{y}} + \frac{\partial C u_z}{\partial \mathrm{z}}$$

Por otro lado:

$$\nabla \cdot (\mathrm{K}\nabla \mathrm{C}) = \frac{\partial}{\partial \mathrm{x}}(K_x \frac{\partial C}{\partial x}) + \frac{\partial}{\partial \mathrm{y}}(K_y \frac{\partial C}{\partial y}) + \frac{\partial}{\partial \mathrm{z}}(K_z \frac{\partial C}{\partial z})$$

de donde, la ecuación se escribirá de la siguiente forma:

$$\frac{\partial C}{\partial t} + \frac{\partial C u_x}{\partial \mathrm{x}} + \frac{\partial C u_y}{\partial \mathrm{y}} + \frac{\partial C u_z}{\partial \mathrm{z}} = \frac{\partial}{\partial \mathrm{x}}\left(K_x \frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial \mathrm{y}}\left(K_y \frac{\partial C}{\partial y}\right) + \frac{\partial}{\partial \mathrm{z}}\left(K_z \frac{\partial C}{\partial z}\right) + S$$

Puesto que se trata de una ecuación en derivadas parciales evolutiva, será necesario establecer el dominio en el cual tendrá validez Ω, así como las condiciones iniciales y de frontera.

Debido a la complejidad que supone resolver la ecuación anterior, en nuestro caso, emplearemos soluciones particulares de la misma que se obtienen realizando una serie de hipótesis que simplifican el modelo.

La complejidad del modelo dependerá de las suposiciones tomadas para restringir el modelo original. Cuanto más restrictivas sean, más simple será el resultado obtenido. Tomando esto como punto de partida, podemos establecer una jerarquía de modelos de difusión, siendo los más simples, como el modelo *Gaussian Puff*, los que están basados en soluciones analíticas (serán los que usemos en nuestro caso).

Para deducir el modelo *Gaussian Puff*, son necesarias las siguientes hipótesis:

1. Supondremos que el término fuente se puede expresar como $S(x,t) = Q\delta(t)\delta(x)\delta(y)\delta(z-H)$, donde $Q\ (kg)$ es la cantidad de contaminante emitido en un punto determinado a una altura H($\vec{x} = (0,0,H)$) y δ es la delta de Dirac:
$$\delta(\mathrm{x}) = \begin{cases} 1 \text{ si } \mathrm{x} = 0 \\ 0 \text{ si } \mathrm{x} \neq 0 \end{cases}$$
Por lo tanto:
$$S(x,t) = \begin{cases} Q\ si\ x = (0,0,H)\ y\ t = 0 \\ 0\ si\ x \neq (0,0,H)\ o\ t \neq 0 \end{cases}$$
2. La velocidad del viento es constante y alineada con el eje x, por lo que, $u = (u,0,0), u \geq 0$.
3. $K_x = K_y = K_z = K(x)$, las constantes de difusión son iguales en todos los ejes y solo dependen de x. Por lo tanto:
$$\nabla \cdot (\mathrm{K}\nabla \mathrm{C}) = \frac{\partial}{\partial \mathrm{x}}(K_x \frac{\partial C}{\partial x}) + \frac{\partial}{\partial \mathrm{y}}(K_x \frac{\partial C}{\partial y}) + \frac{\partial}{\partial \mathrm{z}}(K_x \frac{\partial C}{\partial z})$$
4. Se desprecian los efectos difusivos en el eje x (convección dominante).
$$\frac{\partial}{\partial \mathrm{x}}\left(K_x \frac{\partial C}{\partial x}\right) \cong 0$$
5. Se desprecian las variaciones topográficas y se supone que el suelo se encuentra en $z = 0$.
6. El contaminante no penetra en el suelo:
$$K\frac{\partial C}{\partial z}(x,y,0) = 0$$
7. El dominio de estudio es $\Omega = [0,\infty) \times (-\infty,\infty) \times [0,\infty)$.
8. La fuente es instantánea (en el instante $t = 0$).

En base a las hipótesis enunciadas anteriormente, podemos obtener una solución particular de la ecuación general mediante la transformada de Laplace, que es conocida como *Gaussian Puff* [11]. Según este modelo, si el punto de emisión de un agente químico P_s está situado en $(0,0,H)$, la concentración en cada punto (x,y,z) y en cada instante de tiempo t viene dado por la siguiente expresión:

$$c(x,y,z,t) = \frac{Q_s}{8(\frac{\pi\sigma^2}{2})^{3/2}} \exp\left(-\frac{(x-ut)^2}{2\sigma_x^2}\right) \exp\left(\frac{-y^2}{2\sigma_y^2}\right) [exp\ (-\frac{(z-H)^2}{2\sigma_z^2}) + \exp(-\frac{(z+H)^2}{2\sigma_z^2})]$$

Donde u es la componente x de la velocidad media del viento, Q_s es la cantidad vertida del agente químico, $\sigma_x, \sigma_y, \sigma_z$ son parámetros de forma que dependen de la variable x y la estabilidad S y σ es el producto de los parámetros de forma. En la referencia [6] podemos encontrar una explicación detallada sobre dichos parámetros.

En el caso de que el punto de emisión se encuentre en $P_s = (x_s, y_s, z_s)$, podemos hacer el siguiente cambio de variable:

$$\tilde{x} = x - x_s$$

$$\tilde{y} = y - y_s$$

$$\tilde{z} = z_s$$

De esta forma, el punto de emisión queda situado en $(0,0,z_s)$ y podemos aplicar la fórmula anterior con $H = z_s$, obteniendo entonces:

$$c(x,y,z,t) = \frac{Q_s}{8(\frac{\pi\sigma^2}{2})^{3/2}} \exp\left(-\frac{(x-x_s-ut)^2}{2\sigma_x^2}\right) \exp\left(\frac{-(y-y_s)^2}{2\sigma_y^2}\right) [exp\ (-\frac{(z-z_s)^2}{2\sigma_z^2}) + \exp(-\frac{(z+z_s)^2}{2\sigma_z^2})]$$

Sabiendo que $\sigma = \sigma_x\sigma_y\sigma_z$ y simplificando, obtenemos finalmente:

$$c(x,y,z,t) = \frac{Q_s}{\sqrt{2\pi\sigma_x^2}\sqrt{2\pi\sigma_y^2}\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{(x-x_s-ut)^2}{2\sigma_x^2}\right) \exp\left(\frac{-(y-y_s)^2}{2\sigma_y^2}\right) [exp\ (-\frac{(z-z_s)^2}{2\sigma_z^2}) + \exp(-\frac{(z+z_s)^2}{2\sigma_z^2})]$$

Esta será la función que emplearemos durante el desarrollo del trabajo para calcular la evolución de la concentración del agente químico, tomándola como base para el problema inverso, problema que enunciamos en el siguiente apartado.

3.3 Planteamiento del problema

Nuestro objetivo es determinar la localización del punto de emisión de un agente químico y la cantidad vertida mediante una red de sensores. De esta forma, el primer paso será analizar si existe solución al problema planteado mediante una serie de herramientas matemáticas. Para ello, consideraremos el escenario mostrado en la siguiente figura:

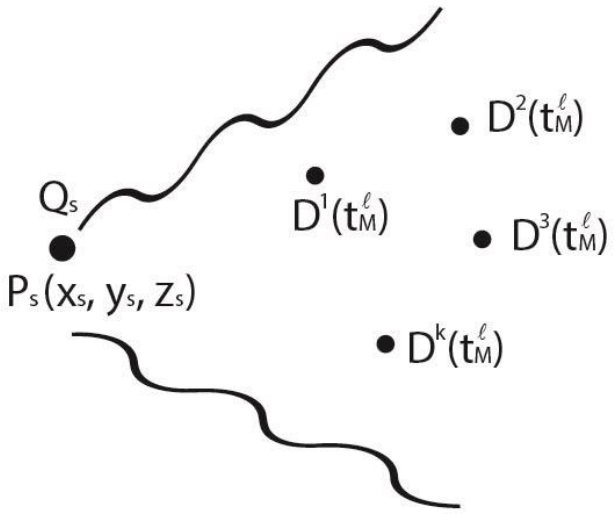

Figura 3-1 Escenario para el planteamiento del problema.

Donde, $P_s = (x_s, y_s, z_s)$ es el punto en el que se produce la emisión, Q_s es la cantidad emitida en el instante inicial y $D^k(t_M^l)$ es la cantidad medida en un periodo de tiempo l por el sensor k, para $k = 1,2,\dots,N_P$ (número del punto de medida) y $l = 1,2,\dots,N_t(k)$ (número del tiempo de medida, el número de tiempos de medida puede depender de cada sensor). Esta misma notación será empleada durante el resto del trabajo.

Recordemos el modelo *Gaussian Puff* que desarrollamos en la sección anterior:

$$c(x,y,z,t) = \frac{Q_s}{\sqrt{2\pi\sigma_x^2}\sqrt{2\pi\sigma_y^2}\sqrt{2\pi\sigma_z^2}} \exp\left(-\frac{(x-x_s-ut)^2}{2\sigma_x^2}\right) \exp\left(\frac{-(y-y_s)^2}{2\sigma_y^2}\right) [exp\ (-\frac{(z-z_s)^2}{2\sigma_z^2}) + \exp(-\frac{(z+z_s)^2}{2\sigma_z^2})]$$

Dado un elemento $(x_s y_s, z_s)$ y Q_s, tenemos una función que describe el comportamiento de la nube toxica, que depende de las coordenadas del punto de emisión, de la cantidad emitida y de las coordenadas espacio-temporales del punto en el que estamos calculando la concentración. Para evitar sobrecargar la notación, llamaremos a esta función también c:

$$c: (x_s y_s, z_s, Q_s, x, y, z, t) \in \mathbb{R}^8 \rightarrow c(x_s y_s, z_s, Q_s, x, y, z, t) \in \mathbb{R}$$

Ahora bien, nuestro sensor nos da la masa total depositada en una placa de área A. El agente químico se acumula en dicho sensor a cierta velocidad de deposición (la forma de obtener su valor será desarrollada en la sección 3.5.6 Sobre el cálculo de la velocidad de deposición del agente químico) que denominaremos de ahora en adelante mediante la notación W_{dep}.

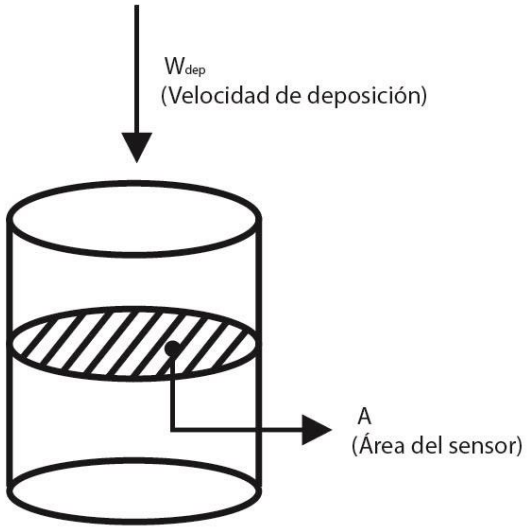

Figura 3-2 Boceto del sensor de medición.

De esta forma, la cantidad medida en un periodo de tiempo t_M^l por el sensor k la denotaremos por $D^k(t_M^l)$. Esta cantidad medida debe ser igual a la predicha por nuestro modelo:

$$\int_0^{t_M^l} A\, W_{dep}\, c(x_s y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t) dt$$

Por lo tanto, nuestro problema consistirá en determinar las coordenadas del punto de emisión (x_s, y_s, z_s) y la cantidad Q_s de forma que la cantidad predicha por el modelo sea igual a la cantidad medida por los sensores:

$$\int_0^{t_M^l} A\, W_{dep}\, c(x_s y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t) dt = D^k(t_M^l) \text{ para } k = 1,2,\dots,N_P \text{ y } l = 1,2,\dots,N_t(k).$$

Desde un punto de vista práctico, es posible que tengamos más medidas que variables a determinar, lo cual se traduce, desde el punto de vista matemático, en un problema sobredeterminado. Esta clase de problemas se pueden resolver empleando mínimos cuadrados (tal y como veremos la sección 3.5.4 Sobre el algoritmo de optimización para la resolución del problema). Sin embargo, un problema de naturaleza ligeramente distinta que conviene estudiar previamente es el de saber cuándo el problema inverso está bien planteado, es decir, cuando podemos determinar de forma única las características del vertido en función de las mediciones tomadas sobre el terreno. Esto ocurre cuando puedo garantizar la existencia de funciones $x_s, y_s, z_{s,} Q_s$ que dependan unívocamente de $x_M^k, y_M^k, z_M^k, t_M^l, c_M^{k,}$ con $k = 1,2,\dots,N_P$ y $l = 1,2,\dots,N_t(k)$. Esto es, puedo determinar de forma única la posición y el término fuente en función de

las medidas tomadas por los sensores. Para abordar el estudio matemático de este problema, emplearemos el teorema de la función implícita [12], tal y como veremos a continuación.

3.4 Análisis matemático del problema

3.4.1 Teorema de la función implícita

La herramienta que nos permite analizar el problema, en base a lo enunciado en la sección anterior, es el teorema de la función implícita. Dicho teorema establece lo siguiente:

Sea Uc$\mathbb{R}^{p+q}$ un entorno de un punto $(a_1,\dots,a_P;b_1,\dots,b_q)$; denotaremos por $x=(x_1,\dots,x_p)$ y por $y=(y_1,\dots,y_q)$, en particular, $a=(a_1,\dots,a_p)\in\ \mathbb{R}^P$ y $b=(b_1,\dots,b_q)\in\mathbb{R}^q$. Sea ahora:

$$\vec{F}\colon U\to\mathbb{R}^q$$

$$(x;y)\to\ \vec{F}(x;y)\in\mathbb{R}^q \text{ tal que:}$$

$$\vec{F}(x;y)=(F_1(x;y),\dots,F_q(x;y))$$

$$\vec{F}\in C^1(U) \text{ (funciones de clase uno en U)}$$

Supongamos ahora que $\vec{F}(a;b)=0$ y que:

$$\det(J_y\vec{F}(a;b))\neq 0$$

Entonces la ecuación $\vec{F}(x;y)=0$ define implícitamente a y como función de clase C^1 de x en un entorno de $x=a$ e $y=b$. Esto es, existe un entorno $\vartheta_a c\mathbb{R}^P$ y una única función $f\colon x\in\vartheta_a\to y=f(x)\in\mathbb{R}^q,\ C^1$, tal que $b=f(a)$ y que satisface a la identidad $F\big(x,f(x)\big)=0$ para $x\in\ \vartheta_a$.

3.4.2 Aplicación del teorema de la función inversa al problema

Veamos en esta sección cómo podemos aplicar el teorema de la función implícita para analizar la existencia de solución de nuestro problema. Para ello, plantearemos nuestro problema para adaptarlo al marco matemático introducido anteriormente.

Supongamos, sin pérdida de generalidad, que el número de tiempos en los que se toman medidas es el mismo en todos los sensores $N_t(k)=N_t$, denotemos por $\overrightarrow{a_s}=(x_s,y_s,z_{s,}Q_s)\in\mathbb{R}^4$ y por $\overrightarrow{b_M}=(x_M^k,y_M^k,z_M^k,t_M^l)\in\mathbb{R}^{3\times N_p\times N_t}$ para $k=1,2,\dots,N_P$ (número del punto de medida) y $l=1,2,\dots,N_t$ (número del tiempo de medida) y definamos $\vec{F}\colon\big(\overrightarrow{a_s};\overrightarrow{b_M}\big)\in\mathbb{R}^4\times\mathbb{R}^{3\times N_p\times N_t}\to\vec{F}\big(\overrightarrow{a_s};\overrightarrow{b_M}\big)\in\mathbb{R}^{N_p\times N_t}$ de la siguiente forma:

$\vec{F}\big(\overrightarrow{a_s};\overrightarrow{b_M}\big)=\int_0^{t_M^l}A\ Wdep\ c(\ x_s,y_s,z_{s,}Q_s,x_M^k,y_M^k,z_M^k,t)dt-D^k\big(t_M^l\big)$ para $k=1,2,\dots,N_P$ y $l=1,2,\dots,N_t$.

Donde la integral de la predicción del modelo nos da un valor teórico de la concentración acumulada del agente químico y $D^k\big(t_M^l\big)$ son las medidas que obtenemos de nuestros sensores. Se trata, por lo tanto, de garantizar la existencia de una cierta función $\vec{f}$ que, dados los datos asociados a los puntos de medida, nos proporcione las características del vertido (punto y cantidad). Esto es, buscamos una función

$$\vec{f}\colon\mathbb{R}^{3\times N_p\times N_t}\to\mathbb{R}^4 \text{ tal que } \vec{F}\big(\vec{f}(\overrightarrow{b_M});\overrightarrow{b_M}\big)=\vec{0}$$

Antes de continuar, debemos observar que, en el caso que planteamos, están intercambiados los papeles de x e y tal y como aparecen en el teorema de la función implícita que hemos enunciado anteriormente. Además, necesitamos imponer la siguiente restricción sobre el número de puntos de medición y tiempos de medida (cuatro medidas para determinar la posición y la cantidad vertida):

$$N_p \times N_t = 4$$

Finalmente, es necesario suponer que, de alguna forma, nuestros sensores han sido calibrados con datos experimentales, lo cual se traduce, desde un punto de vista matemático en la siguiente condición:

$$\exists(\vec{a}_s^0, \vec{b}_s^0) \in \mathbb{R}^4 \times \mathbb{R}^{3\times N_p \times N_t} \text{ tal que}$$

$$\vec{F}\left(\vec{a}_s^0, \vec{b}_s^0\right) = \vec{0}$$

Supondremos también que en el punto de calibración el determinante de la matriz jacobiana es distinto de cero:

$$\det(J_{\vec{a}_s}\vec{F}\left(\vec{a}_s^0, \vec{b}_s^0\right)) \neq 0$$

En base a lo enunciado, podemos considerar cuatro casos iniciales con los que definir un campo vectorial de 4 componentes, dado que buscamos solución a 4 incógnitas. A continuación, veremos cuáles de estos casos cumplen las condiciones que se han enunciado en el teorema de la función implícita.

3.4.3 Escenarios en $\mathbb{R}^4$

En este apartado enunciaremos los diferentes campos vectoriales que forman los escenarios potenciales de resolver nuestro problema, a los que aplicaremos el teorema de la función inversa para determinar si son válidas para obtener una solución.

3.4.3.1 Caso 1: 2 puntos y 2 tiempos

Consideremos los dos puntos de medición $P_M^1(x_M^1, y_M^1, z_M^1)$ y $P_M^2(x_M^2, y_M^2, z_M^2)$ con los que obtenemos dos medidas en cada uno en los tiempos t_M^1 y t_M^2. Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^4$:

$$\begin{pmatrix} F(1) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^1) \\ F(2) = \int_0^{t_M^2} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^2) \\ F(3) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t)dt - D^2(t_M^1) \\ F(4) = \int_0^{t_M^2} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t)dt - D^2(t_M^2) \end{pmatrix}$$

3.4.3.2 Caso 2: 4 puntos y 1 tiempo

Consideremos los 4 puntos de medición $P_M^1(x_M^1, y_M^1, z_M^1)$, $P_M^2(x_M^2, y_M^2, z_M^2)$, $P_M^3(x_M^3, y_M^3, z_M^3)$ y $P_M^4(x_M^4, y_M^4, z_M^4)$, con los que obtenemos cuadro medidas, una por cada punto en el tiempo t_M^1.

$$\begin{pmatrix} F(1) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^1) \\ F(2) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t)dt - D^2(t_M^1) \\ F(3) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^3, y_M^3, z_M^3, t)dt - D^3(t_M^1) \\ F(4) = \int_0^{t_M^1} A\ W dep\ c(x_s, y_s, z_{s,} Q_s, x_M^4, y_M^4, z_M^4, t)dt - D^4(t_M^1) \end{pmatrix}$$

3.4.3.3 Caso 3: 1 punto y 4 tiempos

Consideremos el punto de medición $P_M^1(x_M^1, y_M^1, z_M^1)$ con el que obtenemos cuatro medidas en los tiempos t_M^1, t_M^2, t_M^3 y t_M^4. Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^4$:

$$\begin{pmatrix} F(1) = \int_0^{t_M^1} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^1) \\ F(2) = \int_0^{t_M^2} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^2) \\ F(3) = \int_0^{t_M^3} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^3) \\ F(4) = \int_0^{t_M^4} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^4) \end{pmatrix}$$

3.4.3.4 Caso 4: 2 puntos y 1 tiempo y 1 un punto y 2 tiempos

Consideremos un punto de medición $P_M^1(x_M^1, y_M^1, z_M^1)$ con el que obtenemos dos medidas en los tiempos t_M^1 y t_M^2 y dos puntos que generan otras dos medidas en el tiempo t_M^1. Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^4$:

$$\begin{pmatrix} F(1) = \int_0^{t_M^1} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^1) \\ F(2) = \int_0^{t_M^2} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^1, y_M^1, z_M^1, t)dt - D^1(t_M^2) \\ F(3) = \int_0^{t_M^1} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^2, y_M^2, z_M^2, t)dt - D^2(t_M^1) \\ F(4) = \int_0^{t_M^1} A\, Wdep\, c(x_s, y_s, z_s, Q_s, x_M^3, y_M^3, z_M^3, t)dt - D^3(t_M^1) \end{pmatrix}$$

3.4.4 Análisis de los escenarios en $\mathbb{R}^4$

Una vez que hemos determinado los cuatro posibles casos que se pueden resolver, supuesto disponemos de un punto de calibración, vamos a analizar cuándo podemos garantizar que el determinante de la matriz jacobiana es distinto de cero:

$$\det(J_{\vec{a}_s}\vec{F}(\vec{a}_s^0, \vec{b}_s^0)) \neq 0$$

Recordemos que la matriz jacobiana es una matriz formada por las derivadas parciales de primer orden de una función. Para una función vectorial, supongamos $\vec{F}$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$ es una función que va del espacio euclídeo *n*-dimensional a otro espacio euclídeo *m*-dimensional. Esta función está determinada por m funciones escalares reales:

$y = \vec{F}(x) = (F_1(x), \dots, F_m(x))$

Cuando la función anterior es diferenciable, entonces las derivadas parciales de estas m funciones pueden ser expresadas en una matriz $m \times n$ (la matriz jacobiana de $\vec{F}$):

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

Si $m = n$ entonces la matriz jacobiana es cuadrada y podemos calcular su determinante, conocido como jacobiano. Este determinante en un punto dado nos da información importante sobre el comportamiento de $\vec{F}$ cerca de ese punto.

Para calcular que el $\det(J_{\vec{a}_s}\vec{F}(\vec{a}_s^0, \vec{b}_s^0)) \neq 0$ se ha empleado el programa MATLAB [13] debido al elevado número de cálculos a realizar. Se ha escrito un *script* AI.1 `funcion_inversa.m` para comprobar las condiciones del teorema de la función inversa en nuestro problema.

Comenzamos declarando las variables generales del problema y las variables simbólicas asociadas al término fuente y las coordenadas y tiempos de medición. A continuación, se define la función asociada al modelo de dispersión y los campos vectoriales que componen los cuatro casos que se enunciaron en las secciones 3.4.3.1, 3.4.3.2, 3.4.3.3 y 3.4.3.4. Mediante unos sencillos comandos podemos obtener el determinante de la matriz jacobiana.

Finalizada esta primera versión del script se ha observado que, considerar la función $\vec{F}(\overrightarrow{a_s}; \overrightarrow{b_M}) = \int_0^{t_M^l} A\, W_{dep}\, c(x_s, y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t)dt - D^k(t_M^l)$, consume mucho tiempo de *CPU*, hasta el punto de que el problema resulta prácticamente inabordable empleando MATLAB en un pc estándar, para solventarlo, aproximamos el término $\int_0^{t_M^l} A\, W_{dep}\, c(x_s, y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t)dt$ por el siguiente valor:

$$A\, W_{dep}\, c(x_s, y_s, z_s, Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l$$

De esta forma, ahorramos un enorme tiempo de cálculo que consumía el cálculo de la integral definida. La aproximación anterior constituye una aproximación por exceso del valor real de la integral, lo cual se traduce, desde un punto de vista práctico, en una situación conservadora en la cual se sobreestima el valor real del peligro.

Sin embargo, esta no ha sido la única dificultad que hemos encontrado en el desarrollo inicial del *script* AI.1 `funcion_inversa.m`. A pesar de la primera simplificación que acabamos de introducir, seguimos sin poder obtener un resultado del determinante de la matriz jacobiana, debido a la demanda de cálculo del problema. El único caso capaz de calcular es el de la sección 3.4.3.3 Caso 3: 1 punto y 4 tiempos, para el que obtenemos como resultado que el determinante de la matriz jacobiana es igual a cero, por lo que el teorema de la función implícita no nos permite garantizar que exista una solución, es decir, el sistema no está bien planteado.

En base a esta limitación de cálculo, se reflexiona sobre la necesidad de aplicar medidas que simplificasen el proceso. En un principio se opta por dar valores a las variables generales del problema W_{dep}, A, u y Q_s, pero resultó no ser efectiva para calcular el determinante de la matriz jacobiana. Teniendo en cuenta esta limitación de cálculo, surgió la necesidad de dividir el problema en dos partes:

- Problema 1: determinar la posición (x_s, y_s, z_s) de un vertido químico puntual. Este será el objetivo de nuestro trabajo, dado que el problema 2, enunciado a continuación, ya ha sido desarrollado en el artículo [11].

- Problema 2: determinar la cantidad acumulada o la tasa de emisión de un vertido químico. Se ha considerado de menor interés desarrollarlo por la razón considerada anteriormente.

De esta forma, consideramos la cantidad de agente químico emitido en el instante inicial como un dato conocido, siendo su posición (x_s, y_s, z_s) el nuevo objetivo a resolver.

3.4.1 Escenarios en $\mathbb{R}^3$

Consecuentemente, se replantea nuestro nuevo sistema. Ahora la pregunta es:

$$\exists \vec{f}: \mathbb{R}^{3\times N_p\times N_t} \rightarrow \mathbb{R}^3 \text{ tal que } \vec{F}\left(\vec{f}(\overrightarrow{b_M}); \overrightarrow{b_M}\right) = \vec{0}$$

Imponemos esta vez la restricción:

$$N_p \times N_t = 3$$

Igualmente, sigue siendo necesario que se cumpla que:

$$\exists (\vec{a}_s^0, \vec{b}_s^0) \in \mathbb{R} \times \mathbb{R}^{3\times N_p\times N_t} \text{ tal que}$$

$$\vec{F}\left(\vec{a}_s^0, \vec{b}_s^0\right) = \vec{0}$$

Esto nos reduce un caso respecto a los cuatro propuestos inicialmente, que mostramos a continuación en las siguientes secciones. Además, es necesario que el determinante de la matriz jacobiana sea distinto de cero.

3.4.1.1 Caso 1: 1 punto y 2 tiempos y 1 punto y 1 tiempo

Consideremos un punto de medición $P_M^1(x_M^1, y_M^1, z_M^1)$ con el que obtenemos dos medidas en los tiempos t_M^1 y t_M^2 y un punto que genera otra medida en el tiempo t_M^1.

Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^3$:

$$\begin{pmatrix} F(1) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t_M^1\right) \times t_M^1 - D^1(t_M^1) \\ F(2) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t_M^2\right) \times t_M^2 - D^1(t_M^2) \\ F(3) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t_M^1\right) \times t_M^1 - D^2(t_M^1) \end{pmatrix}$$

3.4.1.2 Caso 2: 3 puntos y 1 tiempo

Consideremos los 3 puntos de medición $P_M^1(x_M^1, y_M^1, z_M^1)$, $P_M^2(x_M^2, y_M^2, z_M^2)$ y $P_M^3(x_M^3, y_M^3, z_M^3)$, con los que obtenemos tres medidas, una por cada punto en el tiempo t_M^1.

Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^3$:

$$\begin{pmatrix} F(1) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^1, y_M^1, z_M^1, t_M^1\right) \times t_M^1 - D^1(t_M^1) \\ F(2) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t_M^1\right) \times t_M^1 - D^2(t_M^1) \\ F(3) = A\, W_{dep}\, c\left(x_s, y_s, z_{s,} Q_s, x_M^2, y_M^2, z_M^2, t_M^1\right) \times t_M^1 - D^3(t_M^1) \end{pmatrix}$$

3.4.1.3 Caso3: 1 punto y 3 tiempos

Consideremos el punto de medición $P_M^1(x_M^1, y_M^1, z_M^1)$ con el que obtenemos tres medidas en los tiempos t_M^1, t_M^2 y t_M^3.

Obtenemos el siguiente campo vectorial $\vec{F} \in \mathbb{R}^3$:

$$\begin{pmatrix} F(1) = A\,W_{dep}\,c\left(x_s, y_s, z_{s,}Q_s, x_M^1, y_M^1, z_M^1, t_M^1\right) \times t_M^1 - D^1(t_M^1) \\ F(2) = A\,W_{dep}\,c\left(x_s, y_s, z_{s,}Q_s, x_M^1, y_M^1, z_M^1, t_M^2\right) \times t_M^2 - D^1(t_M^2) \\ F(3) = A\,W_{dep}\,c\left(x_s, y_s, z_{s,}Q_s, x_M^1, y_M^1, z_M^1, t_M^3\right) \times t_M^3 - D^1(t_M^3) \end{pmatrix}$$

3.4.2 Análisis de los escenarios en $\mathbb{R}^3$

A continuación, seguimos los pasos mencionados en la sección 3.4.4 Análisis de los escenarios en $\mathbb{R}^4$. Recordemos que para que se cumpla el teorema de la función implícita es necesario que el determinante de la matriz jacobiana de la función sea distinto de cero en el puto de calibración, esto es:

$$\det(J_{\vec{a}_s}\vec{F}(\vec{a}_s^0, \vec{b}_s^0)) \neq 0$$

Por lo tanto, se ha adaptado el *script* AI.1 `funcion_inversa.m` en base a las dos modificaciones respecto de los casos iniciales de la sección 3.4.4 Escenarios en $\mathbb{R}^4$, con el objetivo de comprobar las condiciones del teorema de la función implícita en nuestro problema de caracterización del término fuente en un incidente químico. De esta forma, dicho *script* AI.1 `funcion_inversa.m` queda estructurado de la siguiente forma:

- Declaramos las variables generales del problema.
- Declaramos las variables simbólicas asociadas al término fuente.
- Declaramos las variables generales para evaluar el modelo de dispersión.
- Declaramos las variables simbólicas asociadas a las coordenadas de medición.
- Declaramos las variables simbólicas asociadas a los tiempos de medición.
- Declaramos las variables simbólicas asociadas a las mediciones realizadas. Esto es, los tres casos enunciados en la sección 3.4.1 Escenarios en $\mathbb{R}^3$.
- Definimos la función asociada al modelo de dispersión.
- Damos un valor de prueba a las variables σ_x, σ_y y σ_z. Se ha optado por igualar dichos parámetros a $\sigma_x = \sigma_y = \sigma_z = 1$ para simplicar el proceso de análisis de esta fase. Una vez se haya avanzado en el problema, se calcularán rigurosamente según marca la publicación correspondiente [6].
- Inicializamos la función de varias variables. Se escriben los tres campos vectoriales correspondientes a los tres casos de estudio expresados en la sección 3.4.1 Escenarios en $\mathbb{R}^3$ en base a la siguiente expresión:

$F(n) = A\,W_{dep}\,c(x_s, y_s, z_{s,}Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l - D^k(t_M^l)$ para $n = 1,2, \ldots$ (número de medida del sensor), $k = 1,2, \ldots, N_P$ (número del punto de medida) y $l = 1,2, \ldots, N_t(k)$ (número del tiempo de medida).

En el *script*, el campo vectorial correspondiente a cada caso estará compuesto por tres elementos $\vec{F} = \left(F(1), F(2), F(3)\right)$. Esta será la notación empleada para cada caso de dicho *script*.

La expresión $c(x_s, y_s, z_{s,}Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l$ representa la concentración que predice el modelo de dispersión para el punto donde esté ubicado el sensor en cuestión. Recordemos que, por razones de capacidad de CPU, se ha aproximado el valor de la concentración tal y como se explicó en la sección 3.4.4 Análisis de los escenarios en $\mathbb{R}^4$. La expresión $D^k(t_M^l)$ representa la medida tomada por el sensor en cuestión. Precisamente, serán estas dos expresiones las que usaremos para resolver la posición de la fuente de emisión mediante el método de los mínimos cuadrados, que abordaremos en la sección 3.5 Resolución del problema.

A continuación, calculamos la matriz jacobiana de cada función mediante el comando `jacobian` [14] y el determinante de dicha matriz con el comando `det` [15]. Pasemos a exponer los resultados obtenidos en cada caso.

3.4.2.1 Análisis caso 1: 1 punto y 2 tiempos y 1 punto y 1 tiempo

Ejecutamos el *script* AI.1 `funcion_inversa.m` para el caso 1 y obtenemos resultado de su matriz jacobiana. Comprobamos así que las medidas empleadas para reducir la capacidad de cálculo han sido efectivas. Calculamos el determinante de la matriz jacobiana y obtenemos un resultado distinto de cero, no se muestra la expresión obtenida en esta memoria debido a su extensa longitud.

Cabe mencionar el especial interés de este resultado. El teorema de la función implícita nos ha permitido demostrar a priori que con dos sensores que proporcionen tres medidas en total, somos capaces de determinar la situación de la fuente de emisión de un agente químico. Sin embargo, debemos hacer una segunda comprobación ya que, en un principio, podemos intuir la posibilidad de que existan determinados puntos que hagan que el determinante valga cero. Para ello, empleamos el comando `solve` [16] sobre el determinante respecto a las variables de posición del segundo sensor (x_M^2, y_M^2, z_M^2), es decir, fijamos el sensor (x_M^1, y_M^1, z_M^1) para calcular qué valores del segundo sensor (x_M^2, y_M^2, z_M^2) hacen que el determinante de la matriz jacobiana sea igual a cero. Obtenemos como resultado:

$$\vec{v} = \begin{pmatrix} 0 \\ \dfrac{y_M^1 z_s + y_s z_M^1 + y_M^1 z_s \exp(2 z_M^1 z_s) - y_s z_M^1 exp(2 z_M^1 z_s)}{z_M^1 + z_s - z_M^1 \exp(2 z_M^1 z_s) + z_s \exp(2 z_M^1 z_s)} \\ 0 \end{pmatrix}$$

Efectivamente, existe un punto no permitido en el que no podemos garantizar que el problema esté bien planteado, ya que el determinante de la matriz jacobiana sería igual a cero. En base a los resultados obtenidos, vemos que el punto no permitido depende, entre otros factores, de la posición de la fuente de emisión, por lo que, a priori, no podemos determinar este punto. Sin embargo, dado que la coordenada z es igual a cero, simplemente debemos evitar colocar los dos sensores en el mismo plano. En cualquier caso, dicho punto, al ser respecto de una fuente de emisión en concreto, en la práctica difícilmente nos veremos limitados por dicho punto no permitido.

Resulta de gran utilidad la posibilidad de poder obtener solución con solo dos sensores, en caso de que ocurra algún incidente con los demás sensores instalados en la red.

3.4.2.2 Análisis caso 2: 3 puntos y 1 tiempo

Volvemos a ejecutar el *script* AI.1 `funcion_inversa.m` siguiendo los mismos comandos. Obtenemos solución de la matriz jacobiana y su determinante. Efectivamente, es un resultado distinto de cero. De nuevo, no mostramos en esta memoria el resultado del determinante debido a la longitud del resultado. Una vez más, debemos plantearnos si existe un punto no permitido. En este caso contamos con tres sensores, por lo que debemos obtener los puntos no permitidos en el conjunto de todas las combinaciones que generen los tres sensores. Sin embargo, dicho cálculo supera la capacidad del programa MATLAB. Por ello, fijamos la posición de dos de los sensores, (x_M^1, y_M^1, z_M^1) y (x_M^2, y_M^2, z_M^2), y obtenemos una solución para el tercer sensor (x_M^3, y_M^3, z_M^3). Aplicando nuevamente el comando `solve` [16], obtenemos un punto no permitido que por su larga extensión no vamos a mostrar en la memoria. Al igual que ocurre con el caso 1, obtenemos que la coordenada z del posible punto no permitido es igual a cero, así que para este caso también debemos situar los sensores en planos diferentes para asegurar que no estén ubicados en dicho punto no permitido.

3.4.2.3 Análisis caso 3: 1 punto y 3 tiempos

Procedemos de la misma forma que en los casos anteriores y, como se podría intuir de manera sencilla, obtenemos que el determinante de la matriz jacobiana es igual a cero. Efectivamente, un único sensor no es capaz de obtener la posición de vertido del agente químico, aunque dispongamos de tres o más medidas de la concentración.

Una vez analizados los tres casos, pasemos a describir el método desarrollado para obtener solución del problema.

3.5 Resolución del problema

Demostrado ya mediante el teorema de la función implícita que podemos obtener solución del problema planteado, reflexionamos sobre el cómo obtenerla. Para ello, desarrollamos una serie de *scripts* que pasamos a explicar en las siguientes secciones.

3.5.1 Sobre los datos a introducir

Generamos el *script* AI.2 `datos.m` donde definimos los datos asociados al problema.

Queda estructurado de la siguiente forma:

- Declaramos las variables generales del problema. Dado que estos datos se emplearán en diferentes *scripts*, emplearemos el comando `global` [17]. Normalmente, en el programa MATLAB, cada función tienes sus propias variables locales, separadas de las del resto de funciones. Sin embargo, si varias funciones declaran una variable particular nombradas como `global`, entonces todas comparten una copia de la variable. Cualquier cambio del valor de dicha variable, en cualquier función, también será visible para todas las funciones que la hayan declarado mediante `global`.
- Establecemos los datos asociados al caso de estudio, son los siguientes:
  - $icaso$: para elegir entre los dos casos desarrollados en las secciones 3.4.1.1 Caso 1: 1 punto y 2 tiempos y 1 punto y 1 tiempo y 3.4.1.2 Caso 2: 3 puntos y 1 tiempo, respectivamente. Existe un tercer caso, formado por 4 puntos de medidas y 4 tiempos, para estudiar si la redundancia de medidas mejora notablemente la precisión de las coordenadas espaciales de la fuente de emisión. Cuando hablamos de redundancia, nos referimos a convertir nuestro sistema planteado en sobredeterminado, a diferencia de los dos primeros que plantean un sistema de ecuaciones determinado, igual número de ecuaciones que de incógnitas.
  - A: introducimos el área del recipiente del sensor en el que se acumula el agente químico en m^2.
  - W_{dep}: el valor de la velocidad de deposición será calculado en el *script* AI.6 `vd.m` en base a la publicación [6]. En la sección 3.5.6 Sobre el cálculo de la velocidad de deposición del agente químico se explicará con mayor detalle la forma de obtener dicho parámetro, función de la velocidad del viento u en m/s y la estabilidad atmosférica S.
  - u: componente x de la velocidad media del viento en m/s.
  - Q_s: cantidad vertida del agente químico en kg.
  - S: valor adimensional de la estabilidad atmosférica, según la publicación [10].
  - DHD: acrónimo en inglés para la distancia de riesgo a sotavento (*Downwind Hazard Distance*) en km. Se obtiene en función de las tablas divulgadas en la publicación [10]. Es función del tipo de agente químico, velocidad del viento, volumen de vertido y estabilidad atmosférica para el umbral toxicológico de la escala AEGL-2 (umbrales máximos de exposición) [18].
- Fijamos las coordenadas espacio-temporales de los puntos de medida. La variable $icaso$ nos permitirá diferenciar cada caso. Empleamos el comando `if` [19] para que el sistema lea los datos correspondientes al caso elegido mediante la variable *icaso*, mencionada anteriormente. La notación de cada caso, que también emplearemos en el resto de *scripts*, es la siguiente:
  - Caso 1: 1 punto y 2 tiempos y 1 punto y 1 tiempo.
  - Caso 2: 3 puntos y 1 tiempo.
  - Caso 3: 4 puntos y 4 tiempos.

- Por último, introducimos los valores de un punto de vertido aleatorio para generar medidas sintéticas que simularán las medidas que proporcionaría nuestra red de sensores. En el *script* AI.3 `medidas.m` desarrollaremos dichas medidas sintéticas.

3.5.2 Sobre la generación de medidas sintéticas

En el *script* AI.3 `medidas.m` se ha escrito una función en MATLAB que genera medidas sintéticas que emplearemos en la resolución de nuestro problema simulando las medidas que tomarían los sensores instalados. Representa el dato que denominamos mediante la notación $D^k\left(t_M^l\right)$ para $k = 1,2, \dots, N_P$ y $l = 1,2, \dots, N_t(k)$. Se estructura de la siguiente forma:

- Declaramos las variables globales mediante el comando `global` [17], como ya se explicó en la sección anterior 3.5.1 Sobre los datos a introducir.
- Definimos una serie de variables simbólicas necesarias para evaluar la función de concentración.
- Definimos la función de concentración.
- Generamos las medidas correspondientes a cada caso.
  - En primer lugar, calculamos la concentración del agente químico en kg/m^3 de cada punto en su tiempo correspondiente, sustituyendo los valores necesarios en la función de concentración mediante el comando `subs` [20]. Recordemos que dichos valores fueron ya introducidos en el *script* AI.2 `datos.m`. Obviamente, las medidas sintéticas que obtenemos atienden al modelo de la función de concentración, por lo que son de carácter completamente teórico. En la realidad, las condiciones reales no se ajustarán por completo al modelo y los sensores tomarán medidas en base a cierta tolerancia de error.
  - En segundo y último lugar, calculamos la cantidad acumulada $D^k\left(t_M^l\right)$ de cada sensor en kg. Empleamos la expresión que desarrollamos en la sección 3.4.4 Análisis de los escenarios en $\mathbb{R}^4$:
    $$A\, W_{dep} c\left(x_s, y_s, z_s, Q_s, x_M^k, y_M^k, z_M^k, t_M^l\right) \times t_M^l$$
    para $k = 1,2, \dots, N_P$ (número del punto de medida) y $l = 1,2, \dots, N_t(k)$ (número del tiempo de medida).

3.5.3 Sobre el cálculo de los parámetros de forma σ_x, σ_y y σ_z

La metodología para calcular los parámetros de forma σ_x, σ_y y σ_z, viene divulgada en la publicación [6]. En dicha publicación, los parámetros de forma se calculan según las siguientes expresiones:

$$\sigma_x(x) = F|x|^f$$
$$\sigma_y(x) = F|x|^f$$
$$\sigma_z(x) = G|x|^g$$

Donde x es la distancia a sotavento desde la fuente de emisión y los parámetros F, f, G y g se describen a continuación, para zonas terrestres y velocidad del viento igual o menor a cinco nudos, dado que este será el rango de velocidad del viento que emplearemos en el apartado 4.2 Simulación de prueba para un escenario específico:

$$F = 1.041 + 0.445\, S - 9.395 \times 10^{-2}\, S^2 + 6.322 \times 10^{-3}\, S^3$$

$$f = 0.783 - 5.47 \times 10^{-2}\, S + 1.084 \times 10^{-2}\, S^2 - 7.016 \times 10^{-4}\, S^3$$

Donde S es la estabilidad atmosférica tomando valores desde 1, para muy intestable, hasta 7, para nuy estable, según [10].

A continuación describimos los parámetros G y g, que no dependen de la velocidad del viento.

$$G = 0.1229 \exp(0.3295\ S)$$

$$g = 0.97 - 0.09\ S$$

Por tanto, se ha desarrollado el *script* AI.4 `sigma.m` para calcular el valor de los parámetros. Se estructura de la siguiente forma:

- Declaración de las variables globales, de forma similar a los *scripts* anteriores.
- Definimos la variable simbólica x, que representa la distancia a sotavento desde la fuente de emisión.
- Definimos $F, f, G\ y\ g$, función de la componente x de la velocidad media del viento u y la estabilidad atmosférica S, para zonas terrestres.
- Definimos las funciones de los parámetros de forma:

  $$\sigma_x(x) = F|x|^f$$
  $$\sigma_y(x) = F|x|^f$$
  $$\sigma_z(x) = G|x|^g$$

  Como la función depende de la distancia a sotavento de la fuente de emisión, hemos optado por aproximarlo por el valor medio de dicha función en un intervalo conveniente $[0, L]$. Por lo tanto, los parámetros de forma se expresarán aplicando la fórmula para calcular el valor medio de una función [21] de la siguiente forma:

  $$\sigma = \frac{\int_0^L \sigma(\tilde{x}) d\tilde{x}}{L}$$

  Donde estamos cometiendo el abuso de notación de llamar igual a la función que a su valor medio en el intervalo $[0, L]$. Recordemos que en la fórmula anterior estamos suponiendo implícitamente que el punto de emisión se encuentra ubicado en el punto (0,0,H), por lo tanto, para aplicar la fórmula anterior en nuestro caso, es necesario realizar el siguiente cambio de variable:

  $$\tilde{x} = x - x_s$$

  Supondremos además que $L = DHD$, por lo que, finalmente, la aproximación que realizaremos de los coeficientes de forma obedecerá a las siguientes fórmulas:

  $$\sigma_x = \frac{\int_{x_s}^{x_s+DHD} F\,|x-x_s|^f dx}{DHD}$$
  $$\sigma_y = \frac{\int_{x_s}^{x_s+DHD} F|x-x_s|^f\, dx}{DHD}$$
  $$\sigma_z = \frac{\int_{x_s}^{x_s+DHD} G|x-x_s|^g dx}{DHD}$$

3.5.4 Sobre el algoritmo de optimización para la resolución del problema

En el *script* AI.5 `SolveProblem.m` implementamos las opciones que definen el algoritmo de optimización empleado para resolver el problema. Será esta función a la que llamaremos en MATLAB para obtener solución del problema. Por lo tanto, se estructura para que primero se ejecuten linealmente los diferentes *scripts* necesarios de la siguiente forma:

- Declaramos las variables globales que emplearemos.
- Llamamos a la función de datos AI.2 `datos.m`.
- Generamos los valores de sigma AI.4 `sigma.m`.
- Generamos las medidas sintéticas AI.3 `medidas.m`.
- Empleamos el comando `lsqnonlin` [22] que permite resolver problemas de mínimos cuadrados no lineales. Dicho comando atiende a la siguiente estructura:

```
x = lsqnonlin(fun,x0,lb,ub,options)
```

- Definimos los parámetros de `options`, detallados en la siguiente tabla:

`options=optimset('Jacobian','on','Display','iter','MaxIter,50,'TolFun',1.e-21,'MaxFunEvals',1000)`	
`Jacobian`	Empleamos el jacobiano de la función objetivo para reducir el número de iteraciones necesarias para la convergencia del resultado.
`on`	Activamos el uso de dicho jacobiano, el algoritmo también funciona si no disponemos de la matriz jacobiana.
`Display`	Configuración de la información de salida del algoritmo.
`iter`	Muestra cada iteración por pantalla y el mensaje de fin de iteración.
`MaxIter`	Máximo número de iteraciones permitidas.
`TolFun`	Tolerancia final del valor de la función.
`MaxFunEvals`	Número máximo de evaluaciones de la función permitidas.

Tabla 3-1 Parámetros de optimización del algoritmo empleados.

- Definimos el punto inicial de arranque del algoritmo `x0`.
- Acotamos inferiormente la solución del punto de vertido `lb`.
- Acotamos superiormente la solución del punto de vertido `ub`.

Una vez hemos definido las entradas necesarias para el comando `lsqnonlin`, lanzamos el algoritmo mediante el comando `lsqnonlin(@EvalCoste,x0,lb,ub,options)`. El algoritmo comienza en el punto `x0` y busca un mínimo de la suma de los cuadrados de las funciones descritas en `@EvalCoste`. La función `@EvalCoste` devuelve un vector o matriz de valores, no la suma al cuadrado de los valores. El algoritmo calcula implícitamente la suma de los cuadrados de los componentes de `@EvalCoste`.

3.5.5 Scripts auxiliares al algoritmo de optimización

En esta sección hablaremos sobre los *scripts* auxiliares necesarios para poder lanzar el algoritmo descrito en la sección 3.5.4 Sobre el algoritmo de optimización para la resolución del problema.

3.5.5.1 Sobre la función asociada al algoritmo sin la matriz jacobiana de la función

En el *script* AI.7 `fun.m` se desarrolla la función asociada al problema de los mínimos cuadrados para calcular el error cuadrático en cada uno de los puntos de medición. Se estructura de la siguiente forma:

- Declaración de las variables globales.
- Declaramos la dimensión de la función $\vec{F}$ dependiendo del caso elegido, en base al número de sensores y del número de medidas de cada uno.
- Definimos la función de concentración.
- Calculamos cada concentración asociada a cada medida de los sensores.
- Calculamos las componentes de la función $\vec{F}$ según la expresión:

$$F(n) = A\, W_{dep}\, c(x_s, y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l - D^k(t_M^l)$$

para $n = 1,2,\dots$ (número de medida del sensor), $k = 1,2,\dots, N_P$ (número del punto de medida) y $l = 1,2,\dots, N_t(k)$ (número del tiempo de medida).

3.5.5.2 Sobre la función asociada al algoritmo mediante la matriz jacobiana de la función

Para reducir el número de iteraciones necesarias para obtener una convergencia del resultado calculado por el algoritmo, podemos definir la matriz jacobiana de la función. Esto es, calcular las derivadas parciales de cada componente respecto a las variables x_s, y_s, z_s.

De esta forma, desarrollamos lo siguiente:

- Declaramos las variables globales.
- Declaramos la dimensión de la función $\vec{F}$ dependiendo del caso elegido, en base al número de sensores y del número de medidas de cada uno.
- Definimos la función de concentración.
- Definimos las derivadas parciales de la concentración respecto a las variables x_s, y_s, z_s. En el *script*, hemos denotado estas variables como `xsp,ysp,zsp` para diferenciarlas de `xs,ys,zs` (empleadas para dar valor a las medidas sintéticas). Con el objetivo de aligerar la demanda de cálculo del conjunto, aparecen ya calculadas dichas derivadas. Para calcularas, se ha empleado el *script* AI.9 `calculo_derivadas.m`.
- Calculamos la matriz jacobiana de la función para cada caso de estudio. De esta forma, primero sustituimos las variables de posición del sensor y el tiempo asociado a la medida en las derivadas parciales de la concentración. Finalmente, sustituimos cada valor en el gradiente de la función $\vec{F}$ en cada punto de medición de los sensores:

$$F(n) = A\, W_{dep}\, c(x_s, y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l - D^k(t_M^l)$$
$$\nabla_{(x_s,y_s,z_s)} F(n) = A\, W_{dep}\, \nabla_{(x_s,y_s,z_{s)}} c(x_s, y_s, z_{s,} Q_s, x_M^k, y_M^k, z_M^k, t_M^l) \times t_M^l$$

  De esta manera, definimos la función $J\vec{F}$ con la que lanzamos el algoritmo.

3.5.5.3 Sobre la evaluación de coste

Empleamos este sencillo *script* AI.10 `EvalCoste.m` para que el algoritmo arranque la función y su matriz jacobiana. Recordemos que la matriz jacobiana permite reducir el número de iteraciones necesarias para la convergencia, ya que nos indica la dirección en la cual la función varía más rápidamente.

3.5.6 Sobre el cálculo de la velocidad de deposición del agente químico

En el *script* AI.6 `vd.m` calculamos la velocidad de deposición del agente químico en los sensores. Emplearemos la siguiente fórmula [23]:

$$W_{dep} = \frac{u}{a} \qquad \text{para } L > 0$$

$$W_{dep} = \left(\frac{u}{a}\right) \times \left(1 + \frac{300}{-L}\right)^{2/3} \qquad \text{para } L < 0$$

Donde, a partir de la rugosidad z_0 ($z_0 = 0.02$ para zonas marítimas, mientras que para zonas terrestres $z_0 = 0.2$.) y la velocidad del viento u, se calculan los parámetros necesarios para evaluar la fórmula anterior:

u = velocidad de fricción

$$u = \frac{0.41 \times u}{log\left(\frac{10 + z_0}{z_o}\right)}$$

L = longitud de Monin-Obukhov

$$L = (-0.04 + 0.08 \times (S - 2.5)/3)^{-1}$$

a = parámetro que describe la vegetación

$$a = max\left(100; 500 - 400 \times \frac{z_0 - 0.1}{2}\right)$$

Debemos observar que $1/L$ toma el valor cero para $S = 4$, siendo negativa para valores menores estrictos que cuatro y positiva para valores mayores que cuatro, por lo tanto, a efectos prácticos (para evitar dividir por cero), en el código que hemos desarrollado en MATLAB consideraremos la siguiente modificación de la fórmula anterior:

$$W_{dep} = \frac{u}{a} \qquad \text{para } S > 3.5$$

$$W_{dep} = \left(\frac{u}{a}\right) \times \left(1 + \frac{300}{-L}\right)^{2/3} \qquad \text{resto de los casos}$$

3.5.7 Representación gráfica del penacho del agente químico

Mediante el *script* AI.11 `pinta.m` podemos observar gráficamente la emisión de la nube del agente químico generada sintéticamente y su evolución en función del tiempo. Se trata de una herramienta muy interesante para analizar la posición más conveniente de los sensores. Este análisis se tratará con detalle en la sección 4 Resultados.

4 RESULTADOS

4.1 Introducción

A lo largo de este apartado se mostrarán los resultados obtenidos con el programa desarrollado bajo ciertos escenarios simulados. Recordemos con un sencillo gráfico los pasos que hemos llevado a cabo:

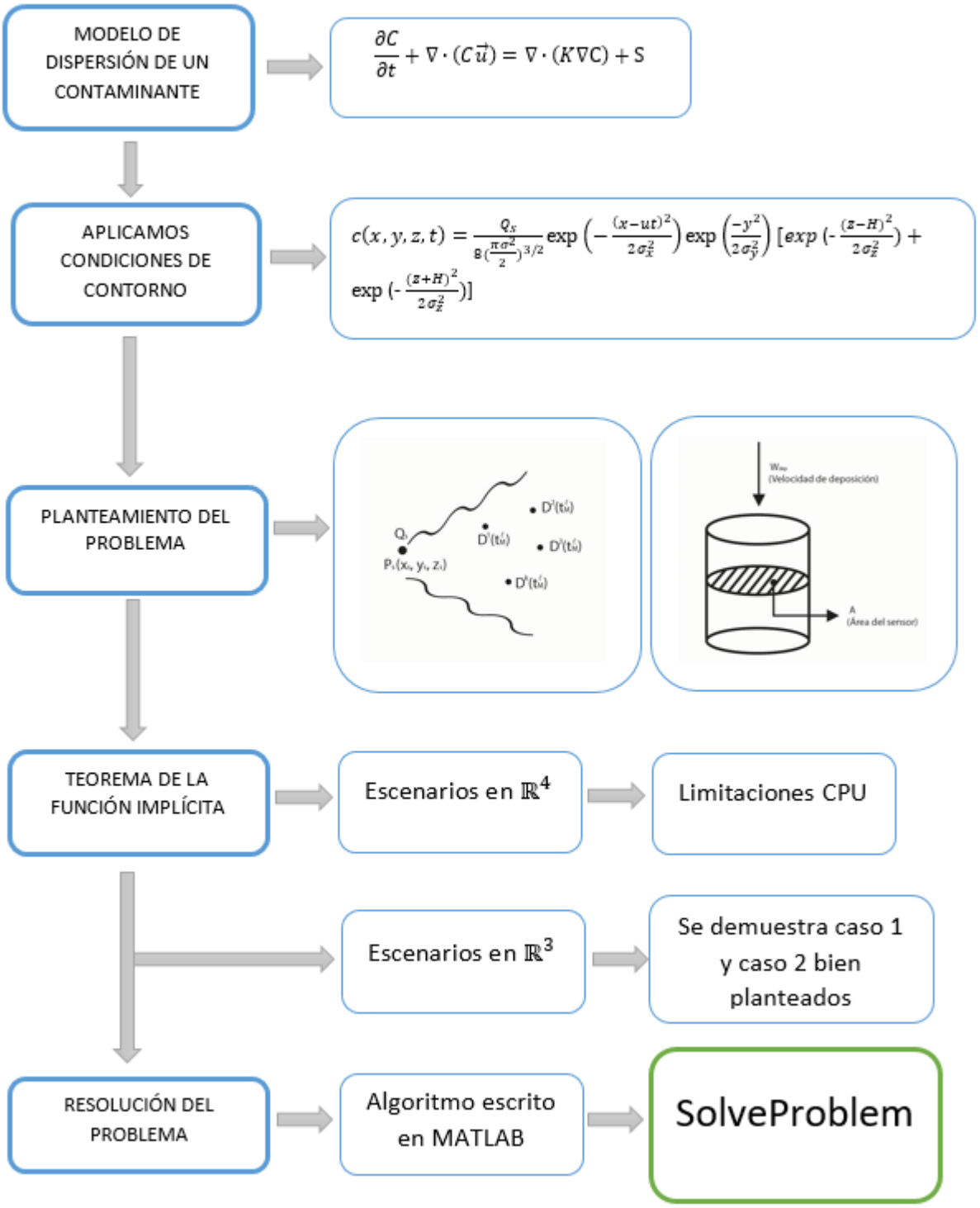

Figura 4-1 Resumen gráfico del apartado 3 Desarrollo del TFG.

4.2 Simulación de prueba para un escenario específico

En este apartado se evaluará el funcionamiento del conjunto de funciones implementadas en MATLAB para un caso en particular de agente químico. Particularmente, hemos escogido el agente VX, por ser uno de los más letales [24]. Los umbrales máximos de exposición que causan efectos severos son, en la escala AEGL (por sus siglas en inglés, *Acute Exposure Guideline Levels*), los siguientes:

$\boldsymbol{Agente\ VX\ 50782-69-9}$					
	10 min	30 min	60 min	4 hr	8 hr
mg/m^3					
AEGL 1	0.00057	0.00033	0.00017	0.00010	0.000071
AEGL 2	0.0072	0.0042	0.0029	0.0015	0.0010
AEGL 3	0.029	0.015	0.010	0.0052	0.0038

Tabla 4-1 Umbrales de exposición del agente VX según escala AEGL [25].

En primer lugar, consultamos la publicación [10] para consultar la distancia de riesgo a sotavento $DHD\ (m)$, en función del agente químico (VX), la velocidad del viento $u\ (m/s)$, la estabilidad atmosférica S y el volumen de vertido $Q_s\ (l)$. Se ha tomado la primera distancia que aparece en la tabla, como punto de partida para el análisis. Podemos verlo en la siguiente figura:

LAND Calculations CAS: 0050782-69-9 , **VX__ , VX**
Threshold : TL based on AEGL2 /
Downwind Hazard Distances [km]

Wind KMPH]		**XS<= 5 [L] < S <= 50 [L] < M <= 200[L] 200[L] < L <= 650 [L] < XL <= 3500 [L]**						
		Stability						
		1	**2**	**3**	**4**	**5**	**6**	**7**
<= 10	**XS**	**1.3**	**1.4**	**1.5**	**1.7**	**1.9**	**2.1**	**2.4**
	S	**2.7**	**3.3**	**4.2**	**5.9**	**7.6**	**11**	**20**
	M	**5.2**	**7.2**	**11**	**22**	**30**	**45**	**65**
	L	**11**	**17**	**30**	**55**	**85**	**135**	**170**
	XL	**40**	**60**	**110**	**240**	**370**	**525**	**535**

Figura 4-2 Tabla extraída de la publicación [10] para obtener la distancia de riesgo a sotavento (DHD).

Para el agente químico VX y una $DHD = 1.3\ km$, se debe cumplir que:

- u $\leq 10\ km/h$ en la dirección positiva del eje x.
- $XS(Q_s)$ $\leq 5\ l$.
- S $= 1$.

Sabiendo que la densidad del agente VX es de $1.008.3\ kg/m^3$ [6] y expresando la velocidad del viento en m/s:

- u $\leq 2.78\ m/s$ en la dirección positiva del eje x.
- $XS(Q_s)$ $\leq 5.0415\ kg$.

Para el área de la placa de deposición de los sensores, tomamos un valor cercano al que aparece en [11] como ejemplo, $A = 0.16\ m^2$.

La velocidad de deposición $W_{dep}\ (m/s)$ se calcula mediante el *script* AI.6 `vd.m` , función de la velocidad del viento $u\ (m/s)$ y la estabilidad atmosférica S.

Conocidos ya los parámetros a introducir y sus valores, se introducen en el *script* AI.2 `datos.m` en las unidades correspondientes, resumidas en la siguiente tabla:

Parámetro	**Símbolo**	**Valor**	**Unidades**
Área del sensor	A	0.16	m^2
Velocidad del viento	u	≤ 2.78	m/s
Cantidad agente químico	Q_s	≤ 5.0415	kg
Estabilidad atmosférica	S	1	Adimensional
Distancia de riesgo a sotavento	DHD	1300	m

Tabla 4-2 Resumen de los parámetros de entrada del *script* AI.2 `datos.m`.

A continuación, definimos la posición de los sensores $x_M^k, y_M^k, z_M^k\ (m)$ para $k = 1,2, \dots, N_P$ y los tiempos de medida $t_M^l\ (s)$ para $l = 1,2, \dots, N_t(k)$. Se introducen en el *script* AI.2 `datos.m`. Recordemos que contamos con 3 casos diferentes para resolver el escenario planteado, como se explica en la sección 3.5.1 Sobre los datos a introducir. También introducimos las coordenadas de una fuente de emisión (x_s, y_s, z_s) simulada para la generación de medidas sintéticas recibidas por la red de sensores, calculadas por el *script* AI.3 `medidas.m`.

Finalmente, introducimos los parámetros relacionados con el algoritmo en el *script* AI.5 `SolveProblem.m`, tal y como se explicó en la sección 3.5.4 Sobre el algoritmo de optimización para la resolución del problema.

Pasemos a evaluar cada caso para el escenario planteado.

4.2.1 Resultados caso 1: 1 sensor y 2 tiempos y 1 sensor y 1 tiempo.

Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(100, 200, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(200, 400, 3)\ m$
	Tiempos de medición	t_M^1	$160\ s$
		t_M^2	$200\ s$

	Coordenadas de la fuente de emisión sintética	(x_s, y_s, z_s)	$(150, 200, 2)\ m$
Script AI.5 `SolveProblem.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(1300, 1300, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000

Tabla 4-3 Parámetros de entrada para el caso 1.

La disposición de los sensores y de la fuente de emisión sería la siguiente:

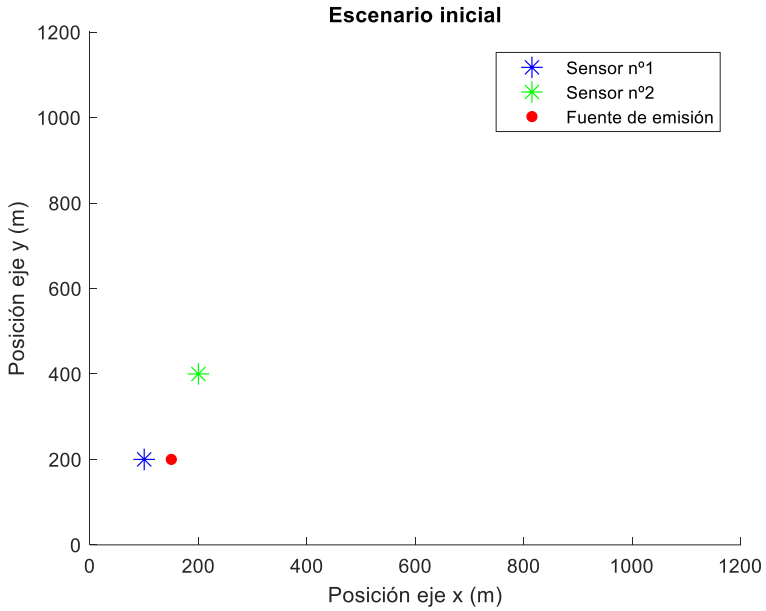

Figura 4-3 Posición de los sensores y de la fuente de emisión para el caso 1.

La nube de agente químico inicial se muestra en la siguiente figura:

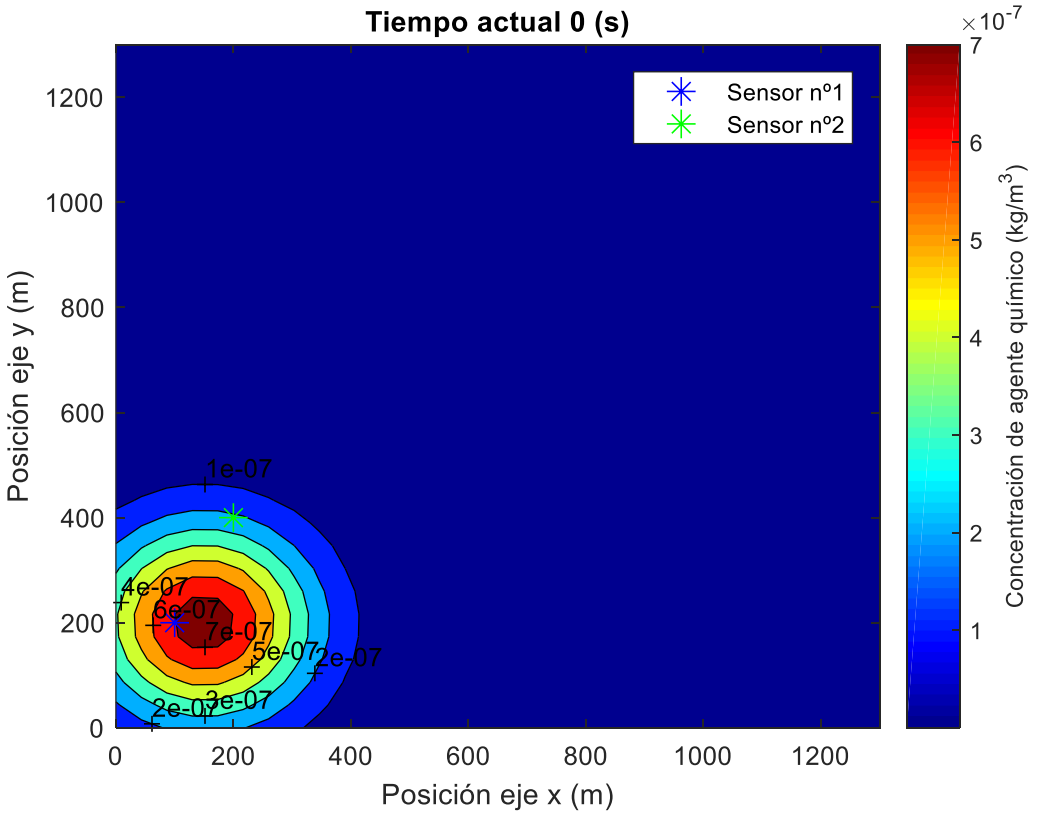

Figura 4-4 Caso 1: concentración del agente químico (kg/m^3) para $t = 0\ s$.

Observemos la evolución de la nube para diferentes instantes de tiempo:

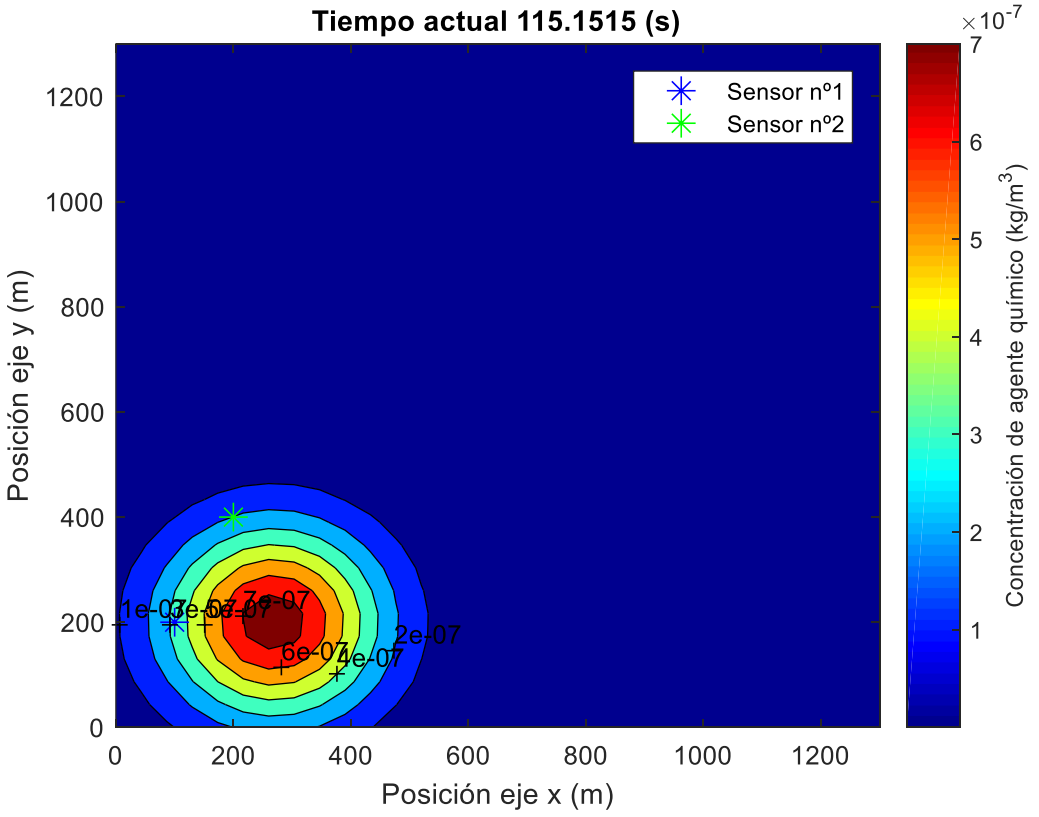

Figura 4-5 Caso 1: concentración del agente químico (kg/m^3) para $t = 115\ s$.

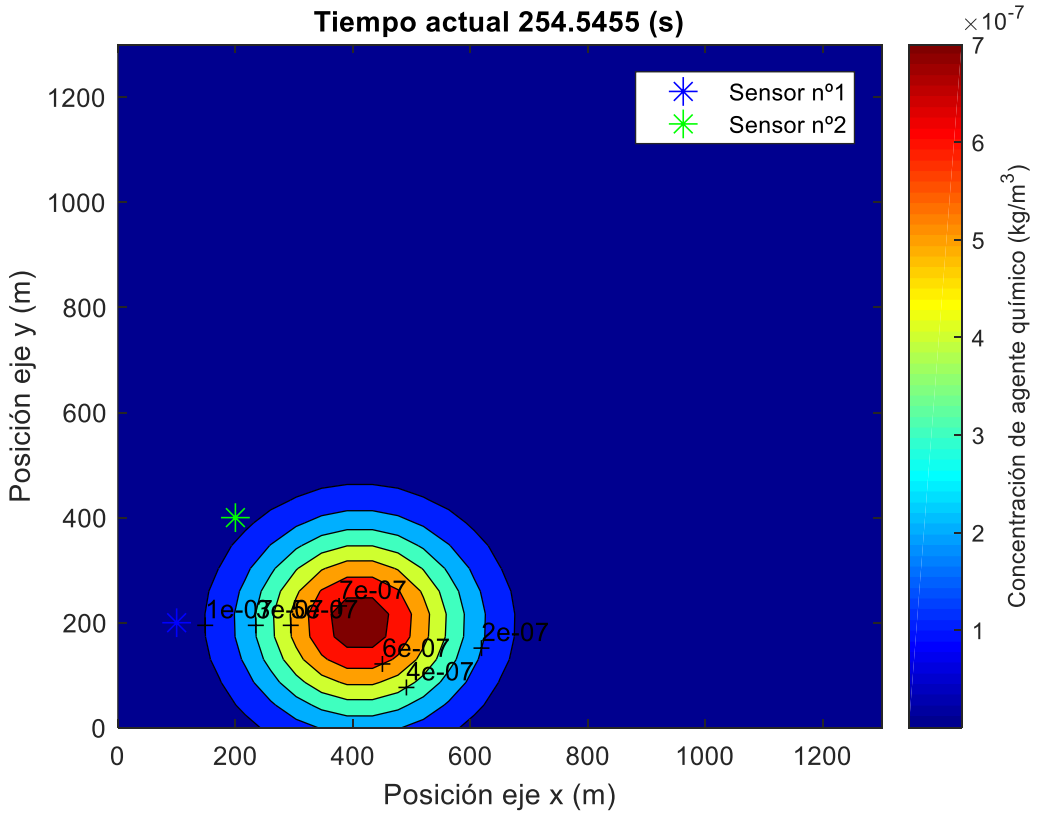

Figura 4-6 Caso 1: concentración del agente químico (kg/m^3) para $t = 254\ s$.

A continuación, lanzamos el algoritmo para obtener solución del problema. Obtenemos los siguientes resultados:

Parámetros de salida		
Solución	x_s	$1.499652415125767e+02\ m$
	y_s	$2.000880548868392e+02\ m$
	z_s	$2.427017037614601e+00\ m$

Tabla 4-4 Parámetros de salida para el caso 1.

Efectivamente, nuestra solución es igual a la fuente de emisión sintética para las coordenadas x_s e y_s. Sin embargo, para la coordenada de altura z_s hay una diferencia considerable. Teniendo en cuenta que las coordenadas x_s e y_s son de un orden muy superior, es lógico que el algoritmo no consiga discriminar con precisión dicha coordenada. De esta forma, a partir de ahora fijaremos nuestra atención en obtener una posición acertada para las coordenadas x_s e y_s, sin tener en cuenta el resultado de la altura z_s.

En el siguiente apartado, analizamos un escenario en el que empleamos tres sensores. Imaginemos que uno de los tres sensores sufra algún tipo de avería, se ha probado que podemos seguir obteniendo solución al problema, suponiendo una correcta distribución de los sensores.

4.2.2 Resultados caso 2: 3 puntos y 1 tiempo.

A continuación, vamos a analizar el caso de tres sensores que proporcionan una medida en un instante determinado.

Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(100, 200, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(200, 400, 3)\ m$
		(x_M^3, y_M^3, z_M^3)	$(300, 250, 5)\ m$
	Tiempo de medición	t_M^1	$160\ s$
	Coordenadas de la fuente de emisión sintética	(x_s, y_s, z_s)	$(150, 200, 2)\ m$
Script AI.5 `SolveProblem.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(1300, 1300, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000

Tabla 4-5 Parámetros de entrada para el caso 2.

La disposición de los sensores y de la fuente de emisión sería la siguiente:

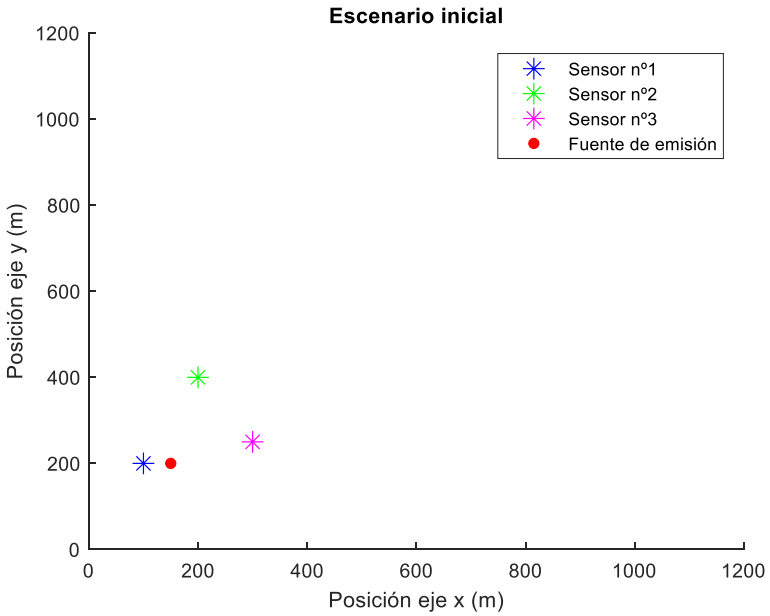

Figura 4-7 Posición de los sensores y de la fuente de emisión para el caso 2.

La nube de agente químico inicial se muestra en la siguiente figura:

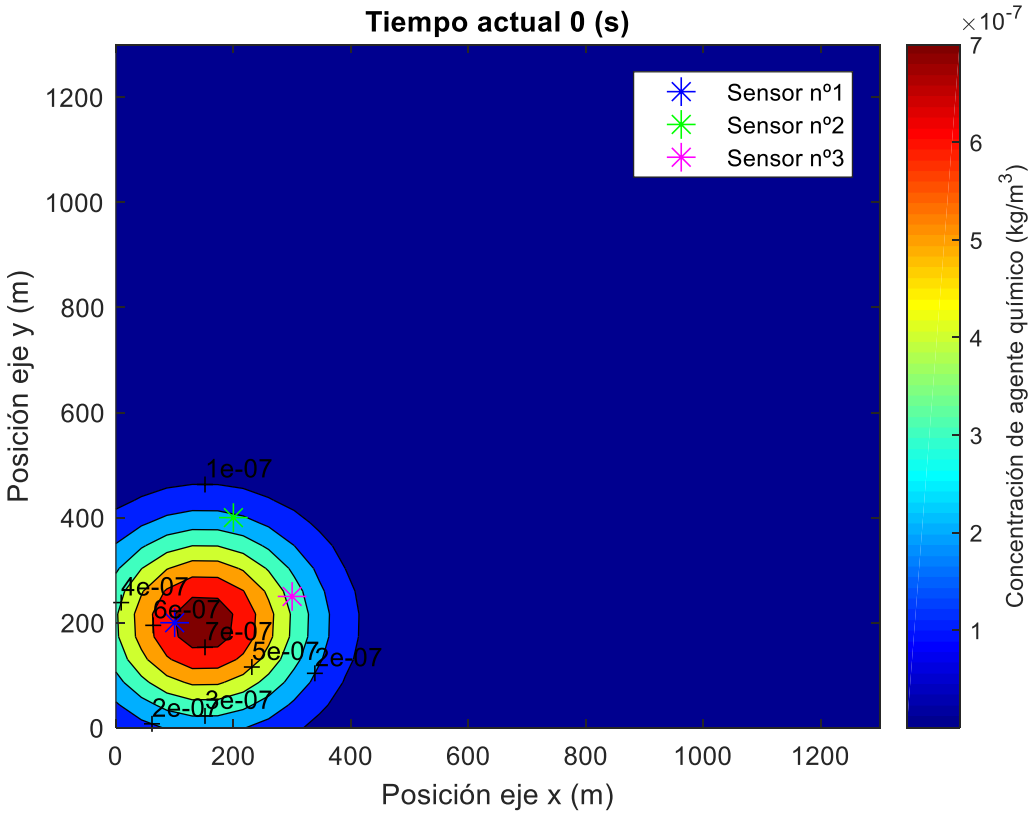

Figura 4-8 Caso 2: concentración del agente químico (kg/m^3) para $t = 0\ s$.

Observemos la evolución de la nube para diferentes instantes de tiempo:

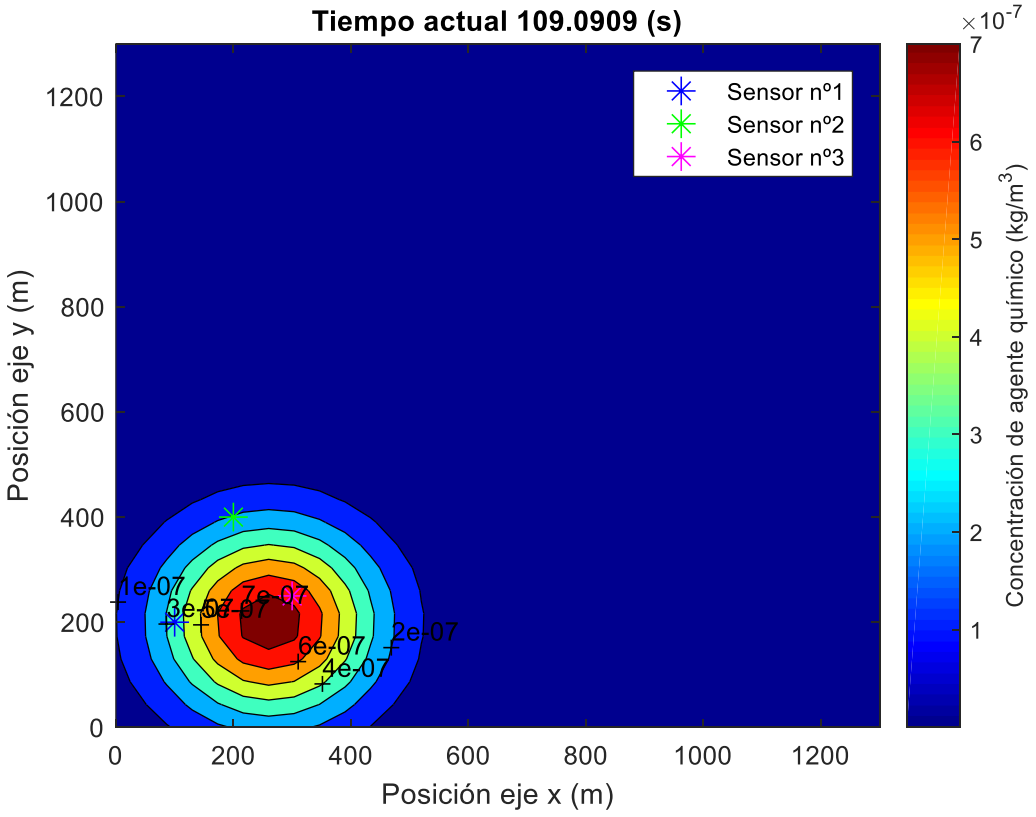

Figura 4-9 Caso 2: concentración del agente químico (kg/m^3) para $t = 109\ s$

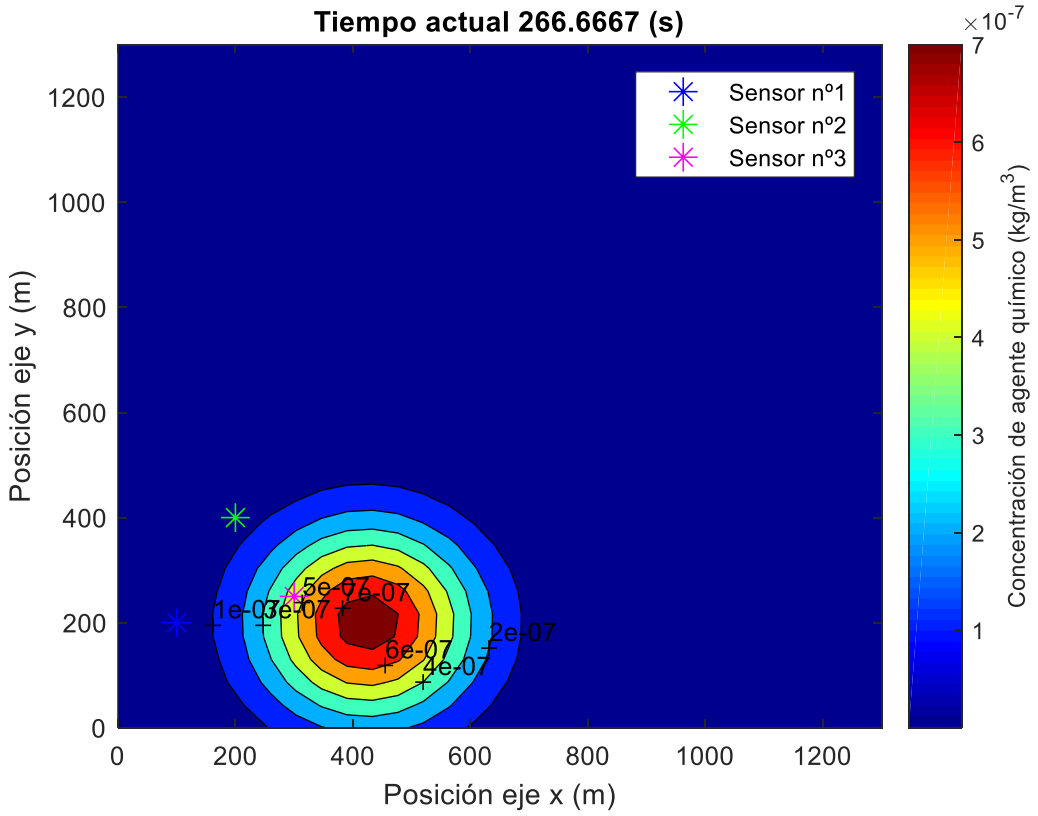

Figura 4-10 Caso 2: concentración del agente químico (kg/m^3) para $t = 266\ s$

A continuación, lanzamos el algoritmo para obtener solución del problema. Obtenemos los siguientes resultados:

Parámetros de salida		
Solución	x_s	$1.500373189360578e+02\ m$
	y_s	$2.001773078607388e+02\ m$
	z_s	$2.344633724156204e+00\ m$

Tabla 4-6 Parámetros de salida para el caso 2.

De nuevo, obtenemos un resultado igual al de la fuente de emisión sintética.

4.3 Análisis de impacto de la posición de la fuente.

Los análisis de la sección 4.2 Simulación de prueba para un escenario específico han sido realizados bajo escenarios completamente definidos, dada una red de sensores fija. Por ello, resulta de interés realizar un mallado de la superficie, lanzado el algoritmo para una posible fuente de emisión en cada uno de los puntos en los que dividimos dicha superficie, variando las coordenadas (x_s, y_s) del agente sintético y manteniendo constante la altura z_s.

De forma gráfica, resumimos los resultados obtenidos en la sección anterior. Posteriormente, se verán los parámetros necesarios para realizar el mallado y sus resultados.

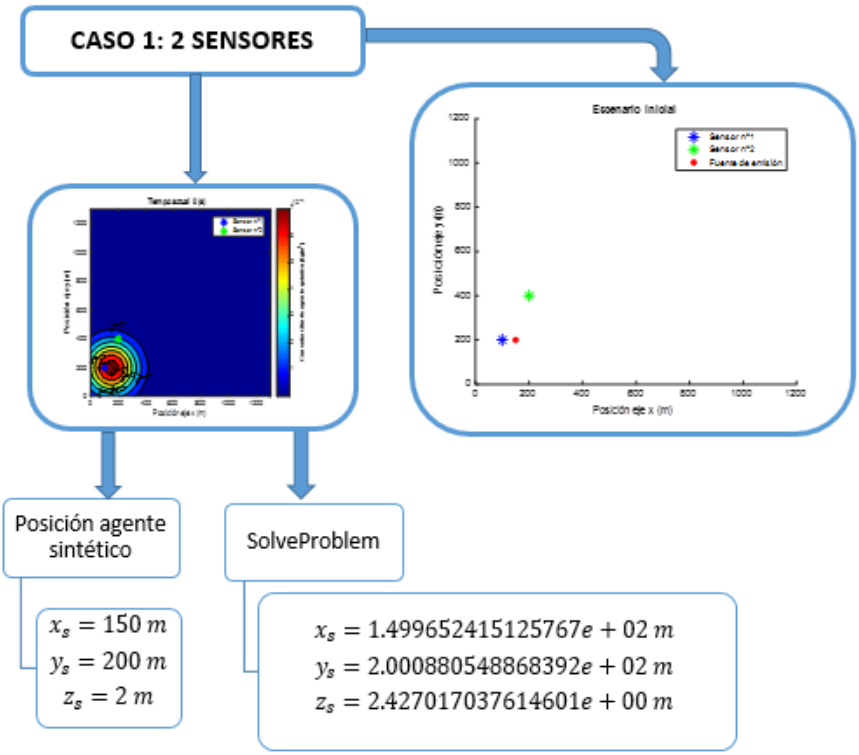

Figura 4-11 Resumen gráfico apartado 4.2.1 Resultados caso 1: 1 sensor y 2 tiempos y 1 sensor y 1 tiempo.

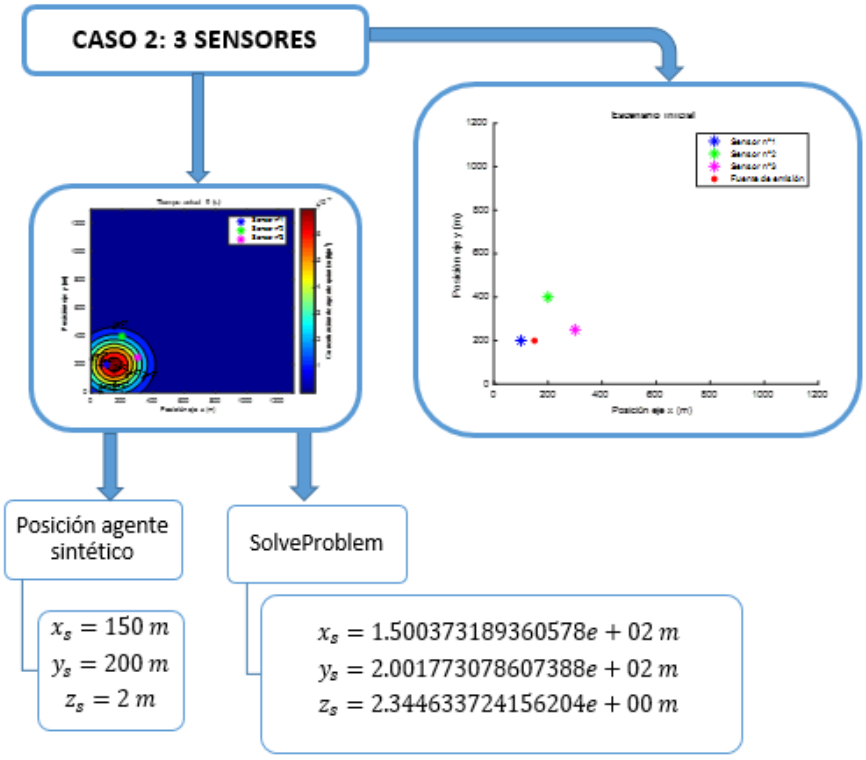

Figura 4-12 Resumen gráfico apartado 4.2.2 Resultados caso 2: 3 puntos y 1 tiempo.

Vamos a analizar el escenario planteado para el caso 2, formado por tres sensores. Para realizar el mallado, empleamos el *script* AI.12 `mallado.m`. Se trata de una pequeña modificación del *script* AI.5 `SolveProblem.m`, al que añadimos un bucle para calcular todos los puntos en los que dividimos el

escenario y los siguientes parámetros, además de los ya explicados en la sección 3.5.4 Sobre el algoritmo de optimización para la resolución del problema:

Parámetro	Símbolo	Unidades
Distancia en el eje $\boldsymbol{x}$	Dx	m
Distancia en el eje $\boldsymbol{y}$	Dy	m
Número de divisiones de los ejes	n	Adimensional

Tabla 4-7 Parámetros adicionales para el mallado de una superficie.

Sobre la posición de la fuente de emisión, variamos en bucle las coordenadas (x_s, y_s) y mantenemos constante la altura z_s, a la que daremos un valor concreto para todos los mallados.

Se han realizado diferentes mallados, que explicamos a continuación:

4.3.1 Primer mallado

Vamos a emplear el escenario planteado para el caso 2 (Figura 4-7), formado por tres sensores.

Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(100, 200, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(200, 400, 3)\ m$
		(x_M^3, y_M^3, z_M^3)	$(300, 250, 5)\ m$
	Tiempo de medición	t_M^1	$160\ s$
	Altura de la fuente de emisión	z_s	$2\ m$
Script AI.12 `mallado.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(1300, 1300, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000
	Sobre el mallado	D_x	$1300\ m$
		D_y	$1300\ m$
		n	100

Tabla 4-8 Parámetros de entrada para el primer mallado.

Lanzamos el algoritmo en bucle y obtenemos las siguientes superficies que representan el error cometido para cada variable espacial:

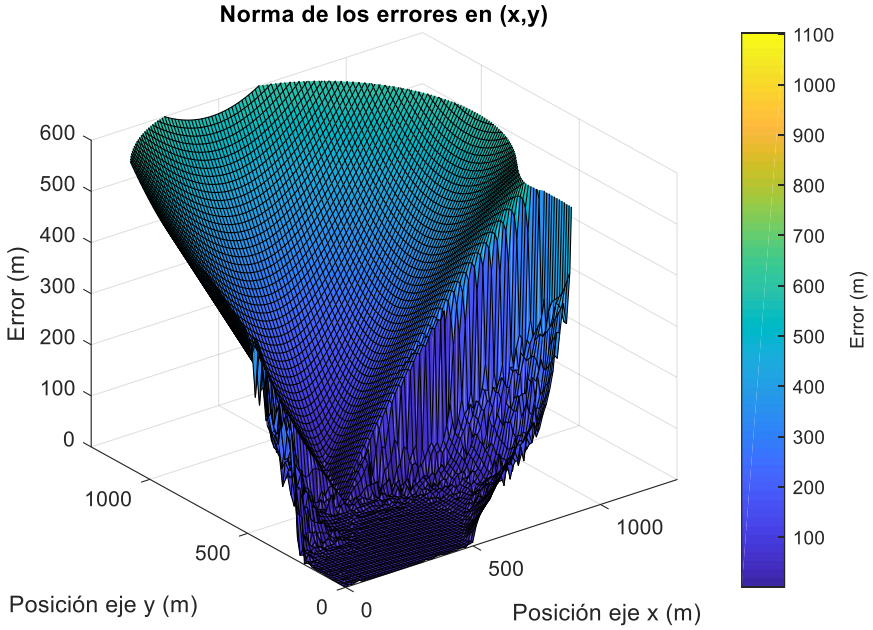

Figura 4-13 Primer mallado: norma de los errores en (x, y).

De manera muy global, observamos que la norma de los errores en (x, y) escala rápidamente a medida que las fuentes de emisión se alejan del área en la que se encuentran los sensores, alcanzando valores que superan los $600\ m$ de error. Gracias a este mallado, encontramos la necesidad de acotar los límites en los que podemos asegurar que el algoritmo es capaz de obtener una solución exacta de la posición del agente químico. También podemos apreciar que para las posiciones cercanas a los sensores, obtenemos soluciones muy precisas. A continuación, analizamos los errores de cada coordenada.

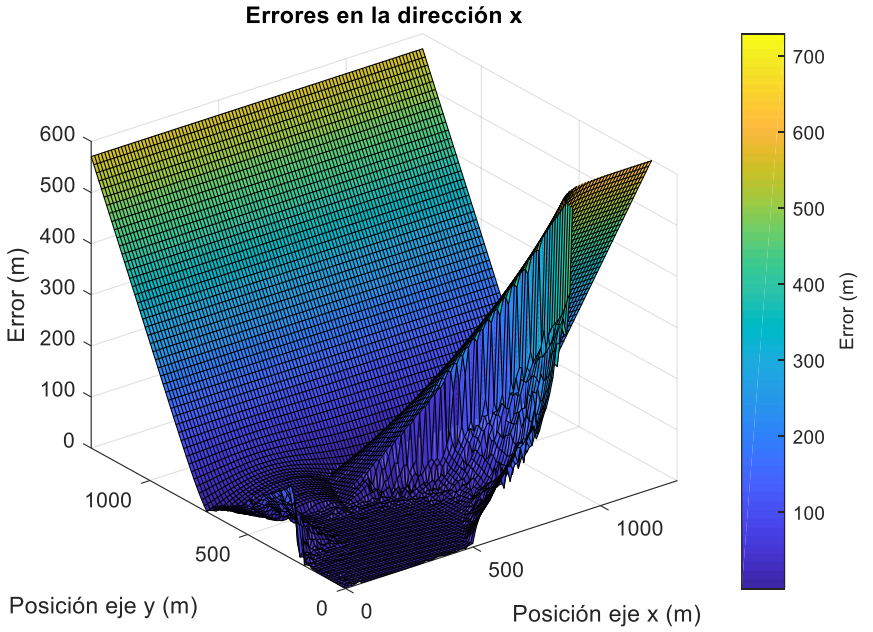

Figura 4-14 Primer mallado: errores en la dirección x.

Vemos que, aproximadamente, los errores dan un salto drástico a partir de los 500 m en el eje x y los 200 m en el eje y.

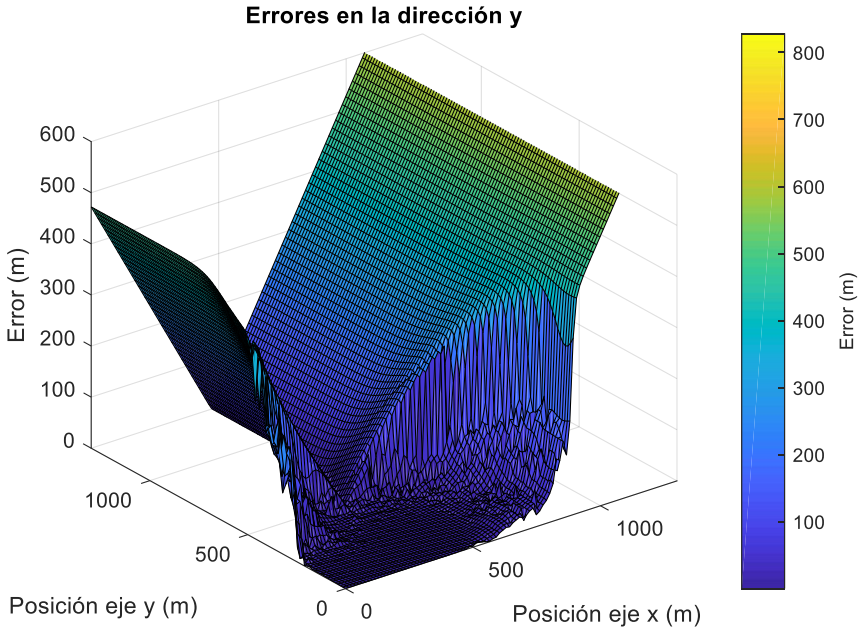

Figura 4-15 Primer mallado: errores en la dirección y.

Los errores escalan de la misma forma que hemos comentado para la coordenada x, en el ejemplo anterior. Se alcanza un error máximo ligeramente superior a los 800 m.

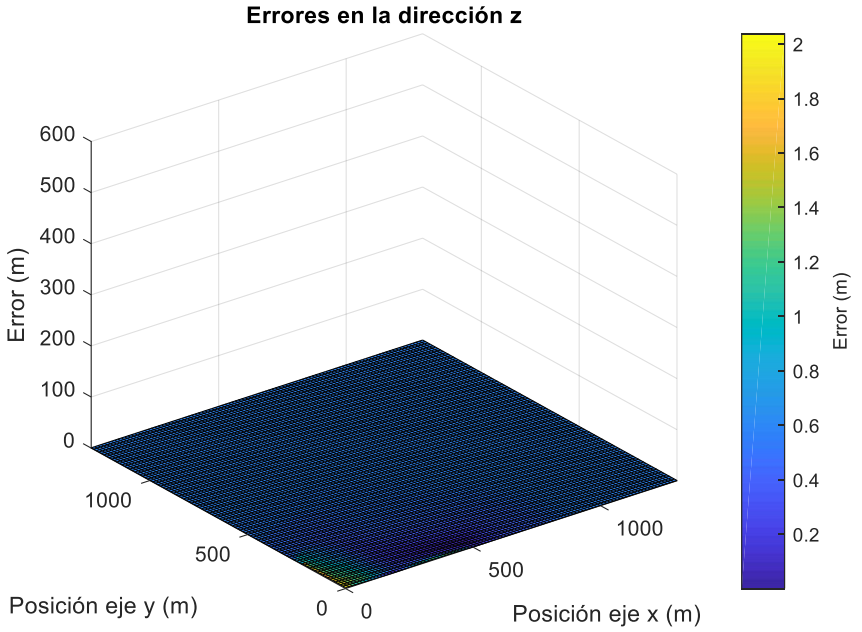

Figura 4-16 Primer mallado: errores en la dirección z.

En cuanto a la altura z, observamos una distribución en casi toda la superficie que no supera los 0.5 m, con máximos ligeramente superiores a los 2 m.

En base a los resultados obtenidos en el mallado anterior, vamos a realizar un segundo mallado bajo los mismos parámetros de entrada, modificando las dimensiones del escenario planteado. Podemos

observar que la zona en la que se encuentran los sensores abarca una zona de errores relativamente bajos, aproximadamente igual a un cuarto de la superficie total. Por ello, realizamos el nuevo mallado abarcando una superficie de $650\ m \times 650\ m$ para las coordenadas x e y, respectivamente.

4.3.2 Segundo mallado

Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(100, 200, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(200, 400, 3)\ m$
		(x_M^3, y_M^3, z_M^3)	$(300, 250, 5)\ m$
	Tiempo de medición	t_M^1	$160\ s$
	Altura de la fuente de emisión	z_s	$2\ m$
Script AI.12 `mallado.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(650, 650, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000
	Sobre el mallado	D_x	$650\ m$
		D_y	$650\ m$
		n	100

Tabla 4-9 Parámetros de entrada para el segundo mallado.

Lanzamos el algoritmo en bucle y obtenemos las siguientes superficies que representan el error cometido para cada variable espacial:

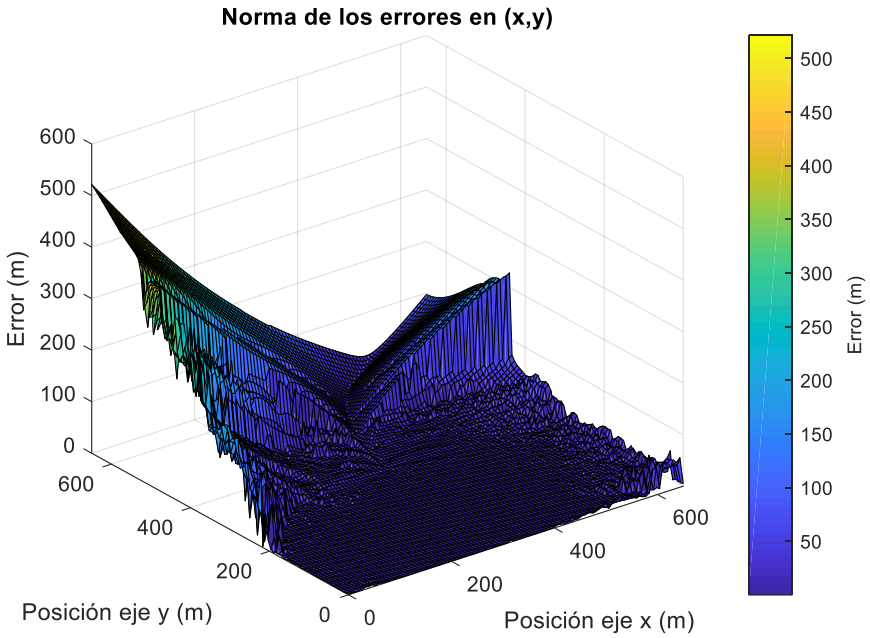

Figura 4-17 Segundo mallado: norma de los errores en (x, y).

En este mallado podemos observar con más detalle las zonas en las que nuestro caso de estudio comete errores demasiado elevados, alcanzando zonas que superan los $500\ m$ de error en la norma de (x, y). Las zonas con menor error corresponden a las que no superan la coordenada de $200\ m$ en el eje y.

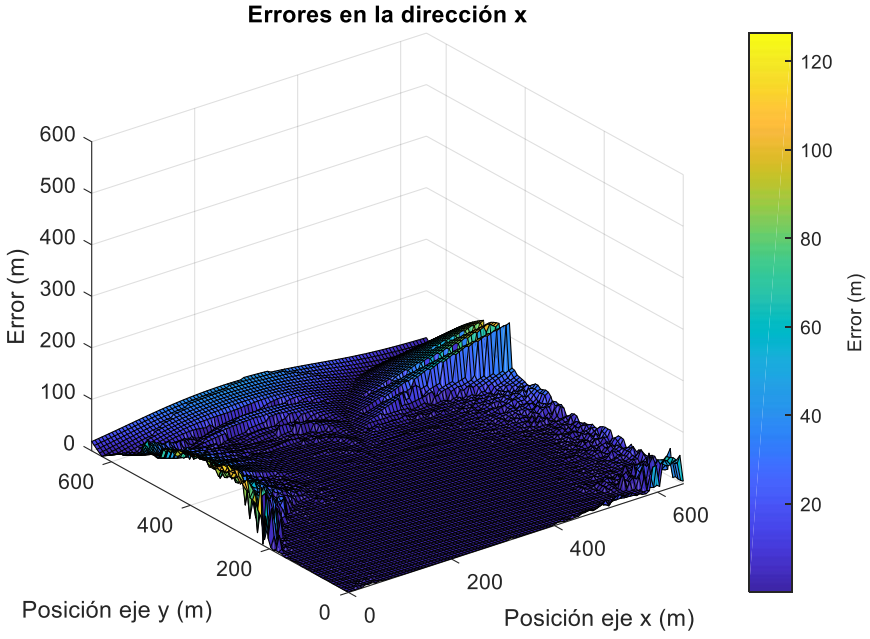

Figura 4-18 Segundo mallado: errores en la dirección x.

Para la coordenada x podemos observar una distribución de errores mucho menor a la de la coordenada y. Se cometen picos de error levemente superiores a los $120\ m$. Para los puntos situados por debajo de los $200\ m$ en la coordenada y, vemos que el error cometido es ínfimo, excepto en la zona posterior a los $500\ m$ en el eje x, cuyo error aumenta ligeramente, pero sin llegar a alcanzar el máximo error presente.

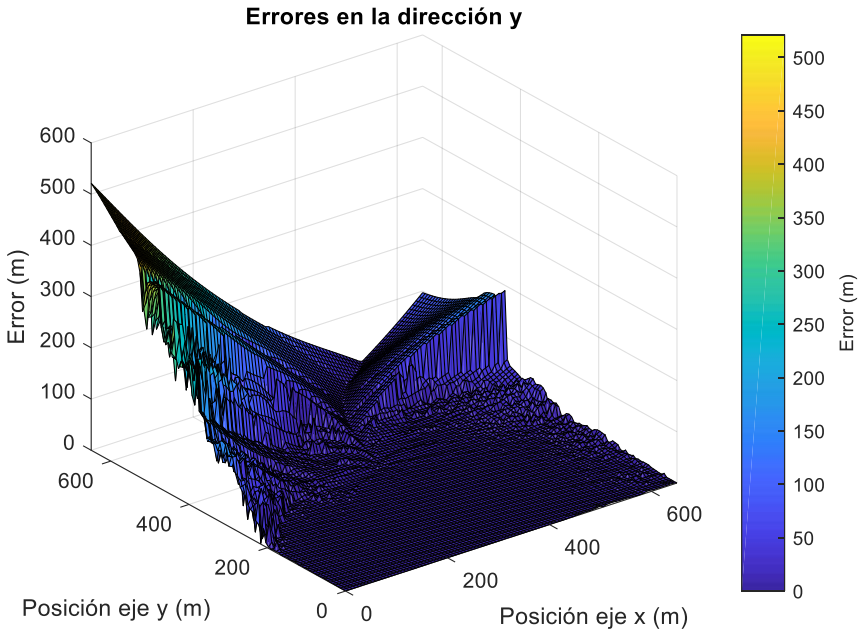

Figura 4-19 Segundo mallado: errores en la dirección y.

Efectivamente, los mayores errores comienzan a partir de los $200\ m$ en el eje y. Aparecen picos que superan hasta los $500\ m$.

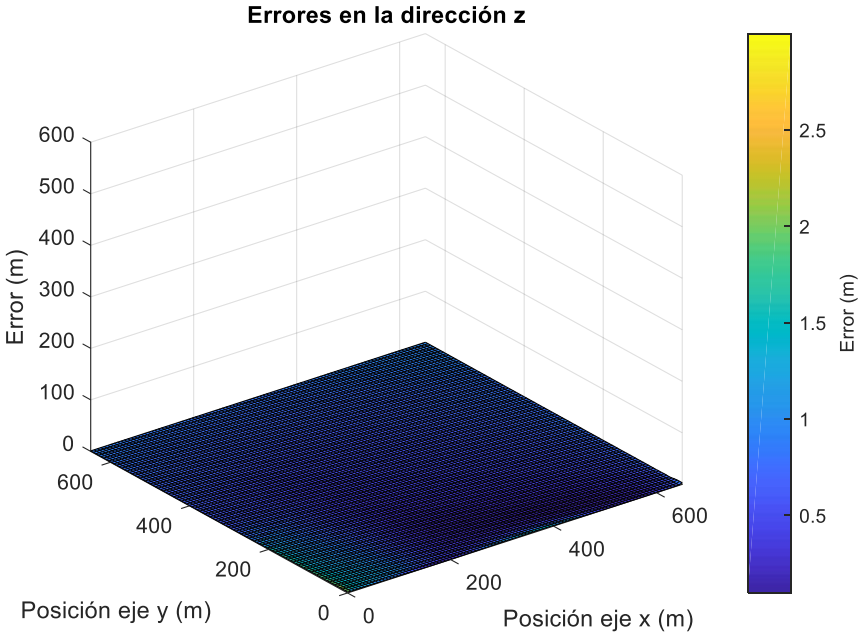

Figura 4-20 Segundo mallado: errores en la dirección z.

En cuanto a la altura z, observamos una distribución en casi toda la superficie que no supera los $0.5\ m$, con un máximo ligeramente superior a los $2.5\ m$.

En base a estos resultados, se demuestra que podemos garantizar resultados de forma exacta para un área de $500\ m \times 200\ m$ para las coordenadas x e y, respectivamente. Dicho esto, vamos a realizar un tercer mallado aumentando en $100\ m$ la distancia en ambas coordenadas, ya que, para dichas distancias, los errores no superan los $40\ m$ para la coordenada x y los $100\ m$ para la coordenada y.

4.3.3 Tercer mallado

En base a los resultados del mallado anterior, vamos a analizar una superficie de $600\ m\ \times 300\ m$, en los ejes x e y, respectivamente. Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(100, 200, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(200, 400, 3)\ m$
		(x_M^3, y_M^3, z_M^3)	$(300, 250, 5)\ m$
	Tiempo de medición	t_M^1	$160\ s$
	Altura de la fuente de emisión	z_s	$2\ m$
Script AI.12 `mallado.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(600, 300, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000
	Sobre el mallado	D_x	$600\ m$
		D_y	$300\ m$
		n	100

Tabla 4-10 Parámetros de entrada para el tercer mallado.

Lanzamos el algoritmo en bucle y obtenemos las siguientes superficies que representan el error cometido para cada variable espacial:

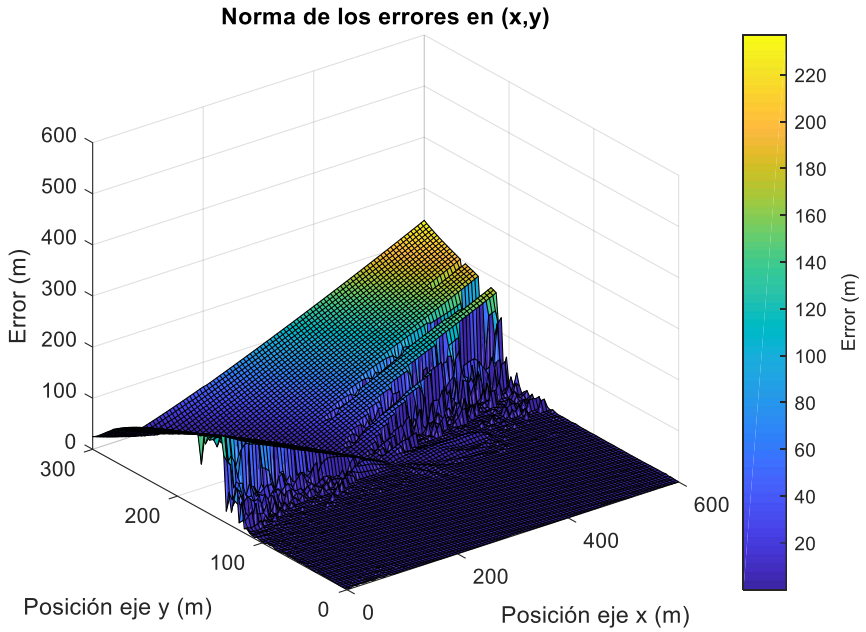

Figura 4-21 Tercer mallado: norma de los errores en (x,y).

Vemos que la zona por debajo de los $100\ m$ en el eje y obtiene buenos resultados, mientras que en en la esquina superior derecha, los errores escalan superando los $220\ m$ de error. Veamos qué componente aporta mayor error de forma global.

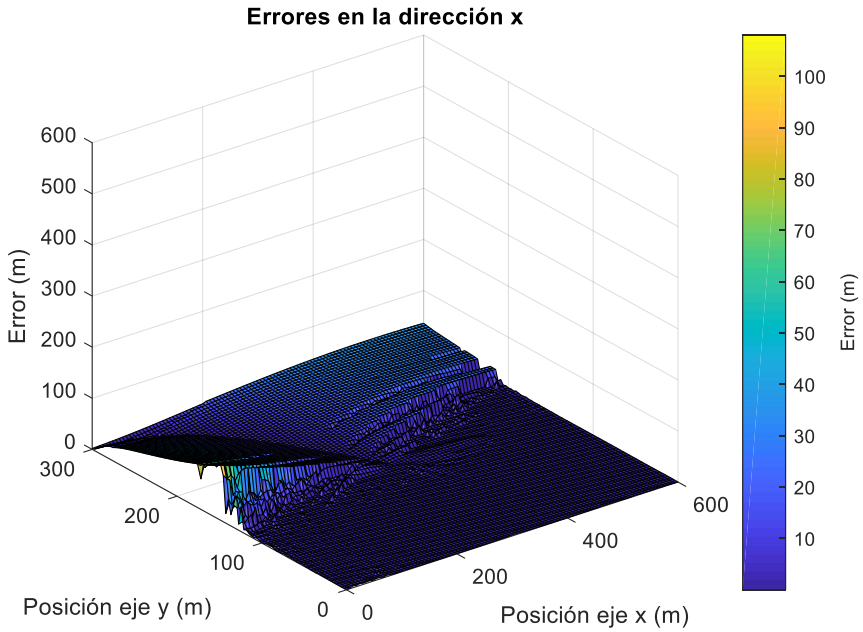

Figura 4-22 Tercer mallado: errores en la dirección x.

Vemos que para la coordenada x, los errores escalan a partir de los $100\ m$ en el eje y, con picos que superan ligeramente los $100\ m$.

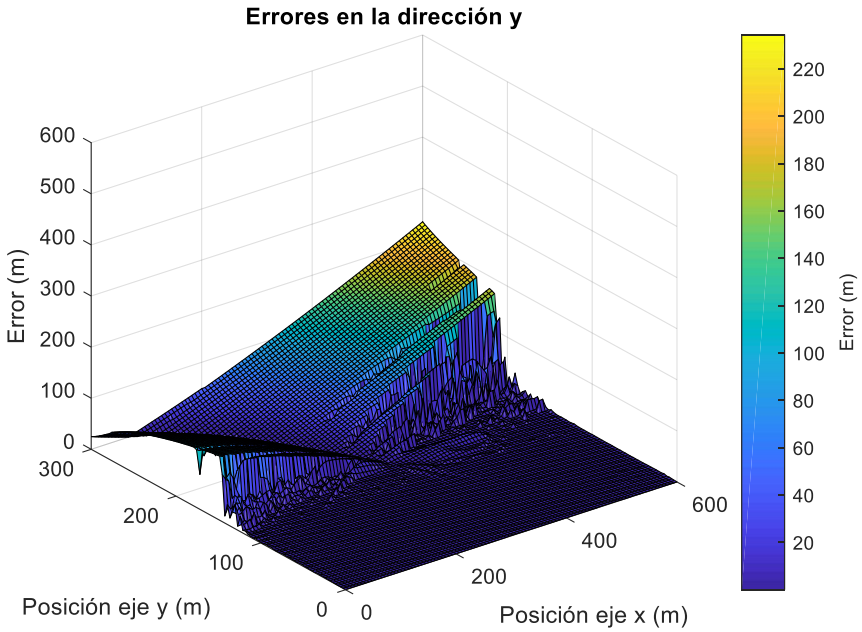

Figura 4-23 Tercer mallado: errores en la dirección y.

Podemos ver claramente que para la coordenada y obtenemos los mayores errores, escalando notablemente hasta superar los $220\ m$ en la esquina superior derecha.

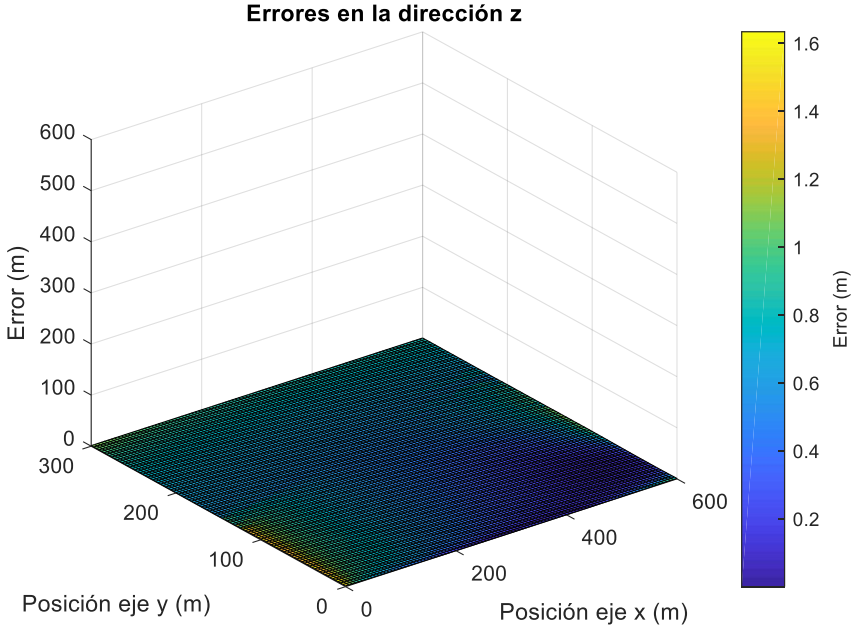

Figura 4-24 Tercer mallado: errores en la dirección z.

Para la altura z, observamos una distribución en casi toda la superficie que no supera los $0.4\ m$, con un máximo ligeramente superior a los $1.6\ m$.

4.3.4 Cuarto mallado

Teniendo en cuenta los resultados obtenidos en el tercer mallado, vamos a añadir un cuarto sensor y aumentamos el número de medidas a cuatro por cada sensor, obteniendo en total dieciséis medidas sintéticas (caso 3), mencionado en la sección 3.5.1 Sobre los datos a introducir.

Con esto buscamos lo siguiente:

- Reducir la zona de mayores errores acumulados en la esquina superior derecha, como se ve en la Figura 4-21 Tercer mallado: norma de los errores en (x,y).
- Comprobar el uso datos redundantes en el algoritmo.

La ubicación de los sensores sería la siguiente:

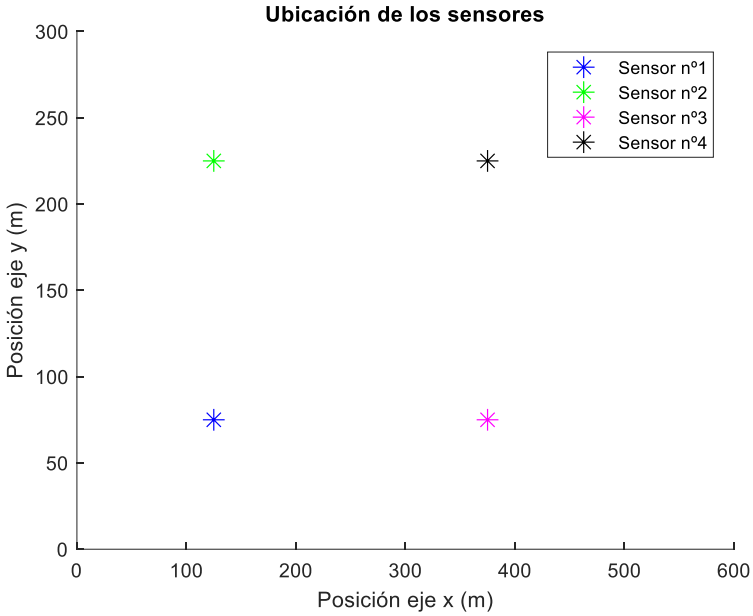

Figura 4-25 Ubicación de los sensores para el cuarto mallado.

Introducimos los siguientes valores:

Parámetros de entrada			
Script AI.2 `datos.m`	Sobre el modelo *Gaussian Puff*	u	$1.02889\ m/s$
		Q_s	$4.0332\ kg$
		S	1
	Sobre los sensores	A	$0.16\ m^2$
		(x_M^1, y_M^1, z_M^1)	$(125, 75, 1)\ m$
		(x_M^2, y_M^2, z_M^2)	$(125, 225, 2)\ m$
		(x_M^3, y_M^3, z_M^3)	$(375, 75, 3)\ m$
		(x_M^4, y_M^4, z_M^4)	$(375, 225, 4)\ m$
	Tiempo de medición	t_M^1	$100\ s$
		t_M^2	$200\ s$
		t_M^3	$300\ s$

		t_M^4	$400\ s$
	Altura de la fuente de emisión	z_s	$2\ m$
Script AI.12 `mallado.m`	Sobre el algoritmo	x_0	$(325, 325, 2.5)\ m$
		l_b	$(0, 0, 0)\ m$
		u_b	$(600, 300, 5)\ m$
		$MaxIter$	50
		$TolFun$	$1e^{-21}$
		$MaxFunEvals$	1000
	Sobre el mallado	D_x	$600\ m$
		D_y	$300\ m$
		n	100

Tabla 4-11 Parámetros de entrada para el cuarto mallado.

Lanzamos el algoritmo en bucle y obtenemos las siguientes superficies que representan el error cometido para cada variable espacial:

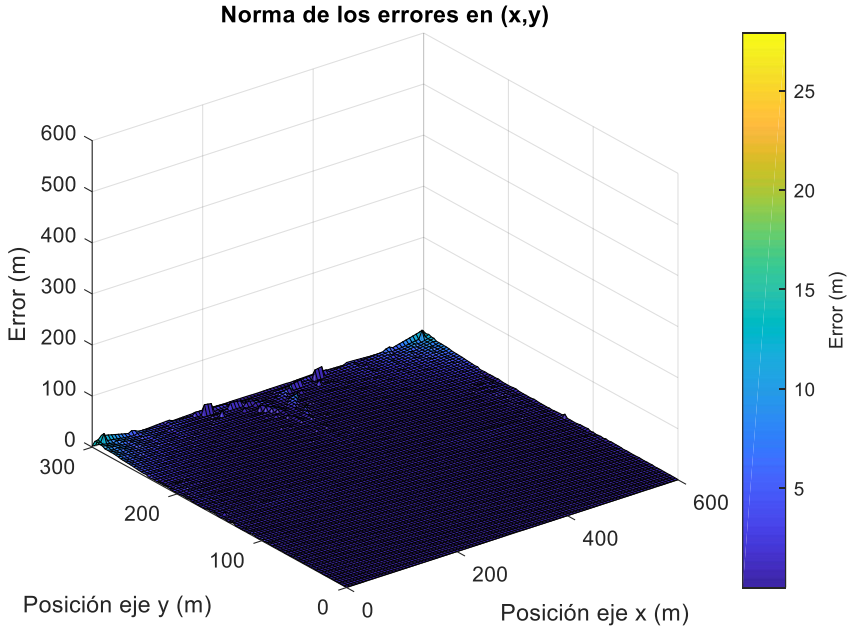

Figura 4-26 Cuarto mallado: norma de los errores en (x,y).

Efectivamente, hemos conseguido reducir notablemente la norma de los errores en (x, y) en prácticamente toda la extensión de la superficie.

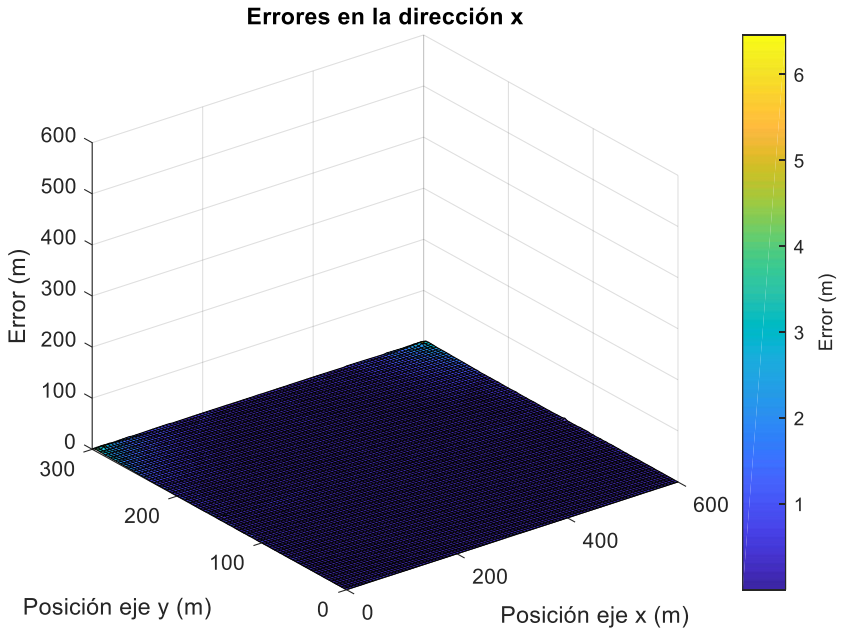

Figura 4-27 Cuarto mallado: errores en la dirección x.

De nuevo, es la coordenada x la que presenta menores errores en comparación con los errores en y.

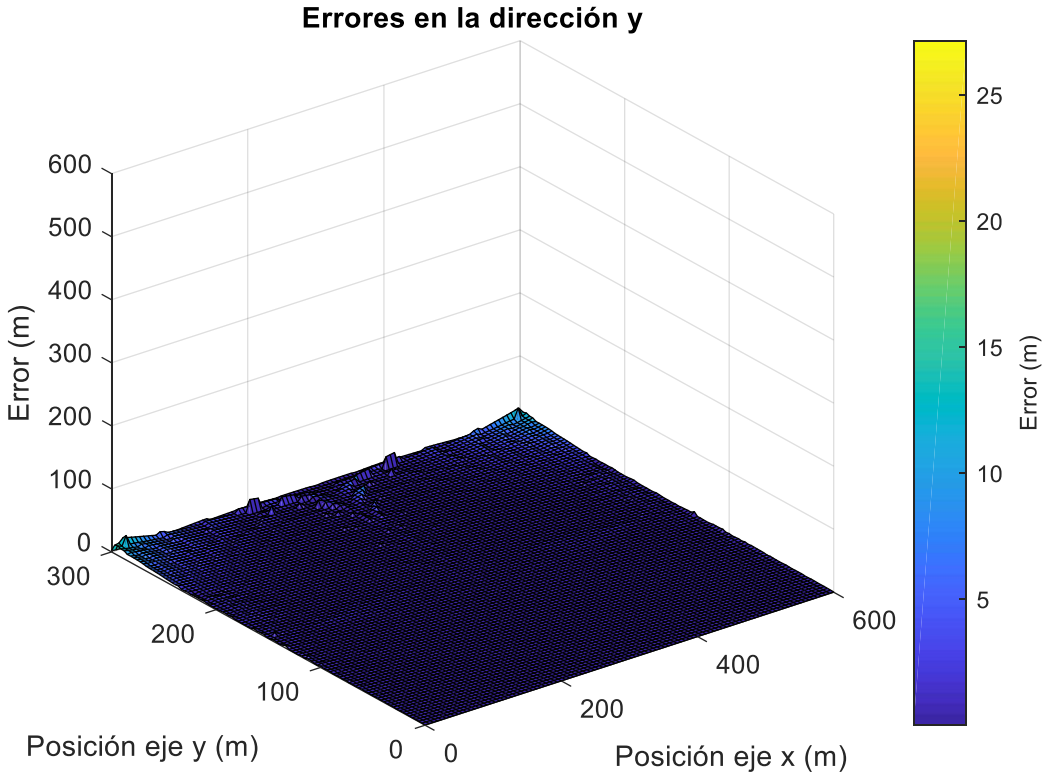

Figura 4-28 Cuarto mallado: errores en la dirección y.

Vemos que la coordenada y, que presentaba los mayores errores para el tercer mallado, se ha reducido en gran medida los errores que presentaba. Sólo se alcanzan picos puntuales sobre los 25 m de error en las esquinas superiores.

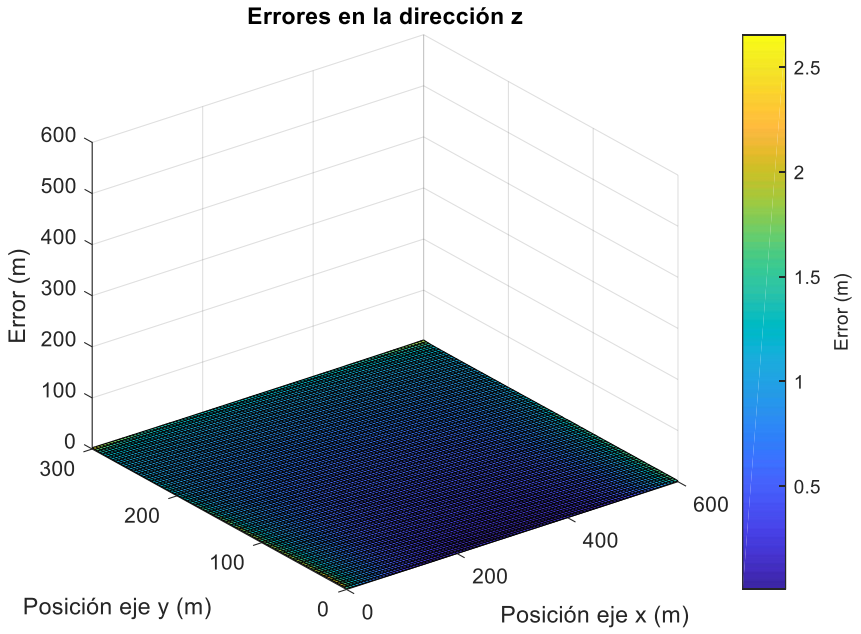

Figura 4-29 Cuarto mallado: errores en la dirección z.

Respecto a la altura z, observamos una distribución en casi toda la superficie que no supera los $0.5\ m$, con un máximo ligeramente superior a los $2.5\ m$.

4.3.5 Conclusiones de los mallados

Tras los cuatro mallados, se ha reducido notablemente el error medio de cada coordenada. Los tres primeros mallados, analizados mediante el caso 2 (formado por tres sensores y una medida por cada sensor), nos han permitido acotar la superficie inicial de $1300\ m \times 1300\ m$, que presentaba errores muy elevados. Tras el tercer mallado, obtenemos una superficie de $600\ m \times 300\ m$, en los ejes x e y, respectivamente, en la que las soluciones obtenidas son exactas. A continuación, partiendo de esta superficie obtenida, realizamos un cuarto y último mallado añadiendo un sensor para reducir el error que se acumula en las esquinas de la superficie y aumentado el número de medidas de cada sensor a cuatro, para evaluar la redundancia de medidas en el algoritmo. Finalmente, obtenemos un valor medio final en la norma de los errores en (x, y) de $0.795\ m$. En la siguiente gráfica podemos observar la disminución de dichos errores tras cada mallado:

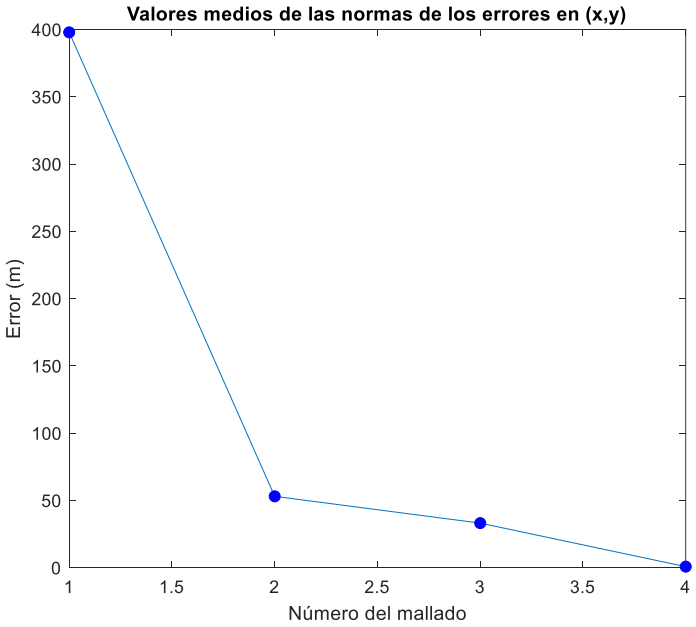

Figura 4-30 Valores medios de las normas de los errores en (x, y) en cada mallado.

En la siguiente figura, se muestra un resumen gráfico del proceso seguido, para facilitar su lectura.

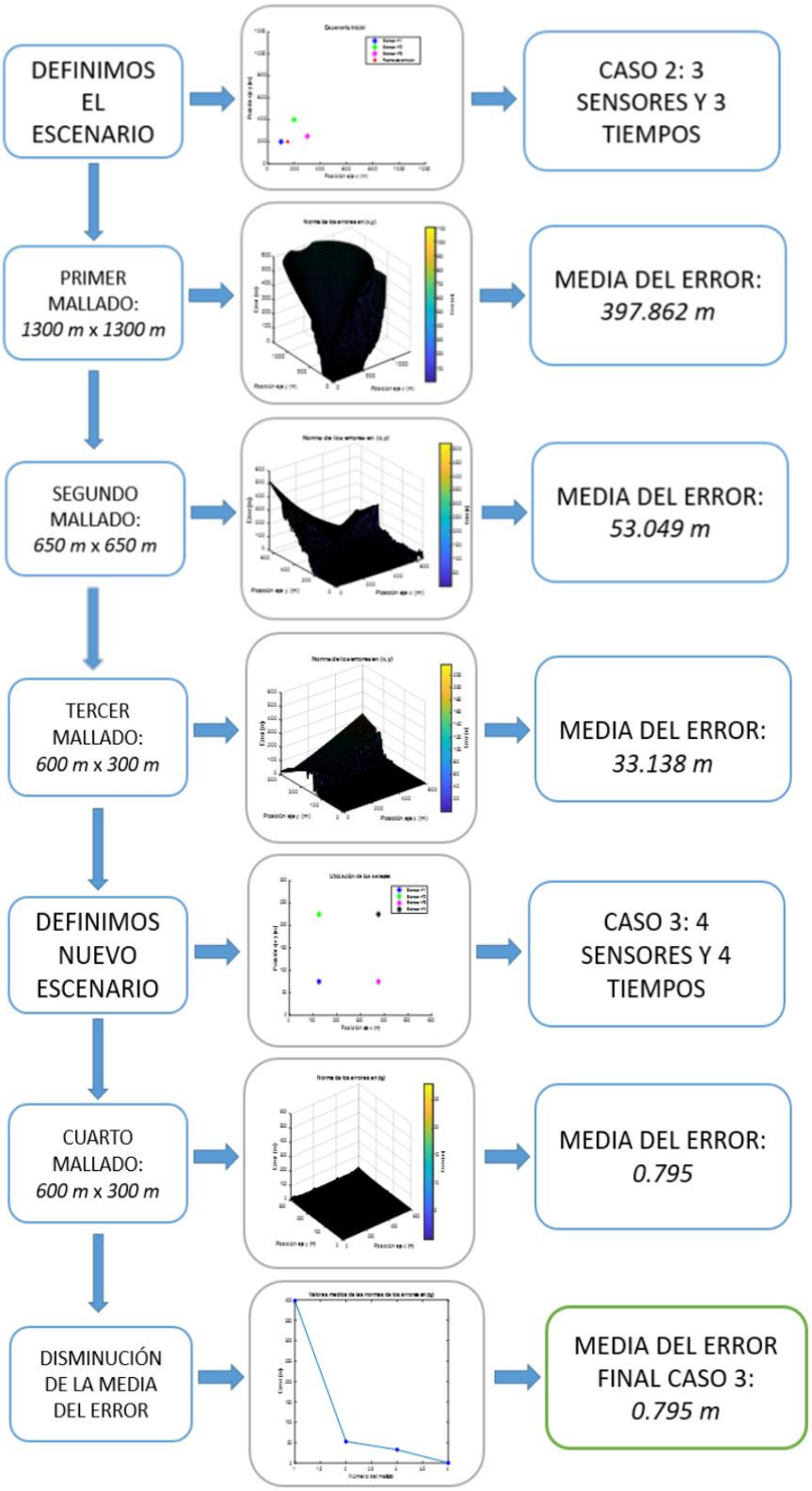

Figura 4-31 Resumen gráfico del apartado 4.3 Análisis de impacto de la posición de la fuente.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Resumen del trabajo desarrollado

A partir del modelo de dispersión de un contaminante, tomando como fuente la publicación [10], hemos aplicado unas hipótesis para obtener una solución particular, denominada *Gaussian Puff*. Durante el desarrollo hemos seguido cuatro bloques de trabajo principales:

1. Primer bloque: analizar la solución particular mediante el teorema de la función inversa para garantizar que nuestro problema esté bien planteado. En base a dificultades relacionadas con la capacidad de cálculo, no podemos analizar la función con las tres variables de posición de la fuente de emisión y la cantidad vertida. Por ello, continuamos nuestro trabajo marcando como objetivo obtener la posición, de mayor interés, tal y como se explica en la sección 3.4.4 Análisis de los escenarios en $\mathbb{R}^4$. Obtenemos dos casos con los que podemos definir bien el problema. Para el primer caso tenemos dos sensores que proporcionan tres medidas en total y para el segundo caso, tres sensores que proporcionan tres medidas en total.
2. Segundo bloque: desarrollar un algoritmo que proporcione solución de la posición. Para ello hemos generado los *scripts* que explicamos en la sección 3.5 Resolución del problema.
3. Tercer bloque: una vez que tenemos los dos casos que cumplen el teorema de la función inversa y un algoritmo para obtener solución, definimos un escenario de vertido del agente químico VX. Para ambos casos obtenemos una solución exacta para las coordenadas (x, y) y una aproximada para la altura z, componente de menor interés.
4. Cuarto bloque: tras los resultados del bloque anterior, se realiza un mallado del escenario para analizar el impacto de la posición de la fuente de emisión. Tomamos el caso 2, formado por tres sensores y se realizan tres mallados bajo las mismas condiciones para acotar un área efectiva en la que la red de sensores pueda abarcar soluciones exactas. Finalmente, realizamos un último mallado añadiendo otro sensor al escenario planteado, dispuestos de forma simétrica. Mediante esto, reducimos las posiciones que presentan un mayor error y comprobamos que la redundancia de medidas sintéticas mejora de forma global los resultados obtenidos. Finalmente, conseguimos acotar un área $600\,m \times 300\,m$ en los ejes x e y, respectivamente, mediante cuatro sensores y dieciséis medidas en total, en el que las soluciones son exactas o presentan un error mínimo.

5.2 Conclusiones y objetivos cumplidos

Tomando como referencia los objetivos de este trabajo, planteados en la sección 1.2 Motivación y objetivos, comentamos las conclusiones obtenidas y los objetivos que se han cumplido.

Respecto al objetivo principal: ≪El sistema desarrollado deberá obtener las coordenadas y la cantidad de la fuente de emisión para un escenario definido por una red fija de sensores instalados previamente≫, teniendo en cuenta las dificultades de demanda de cálculo en el apartado 3.4.4 Análisis de los escenarios en $\mathbb{R}^4$, marcamos como nuevo objetivo desarrollar un algoritmo que obtenga solución de la posición de la fuente de emisión (x_s, y_s, z_s). Los resultados obtenidos tras los análisis mostrados en la sección 4 Resultados muestran resultados exactos para un escenario formado por cuatro sensores y cuatro mediciones por sensor, tal y como mostramos en la siguiente figura:

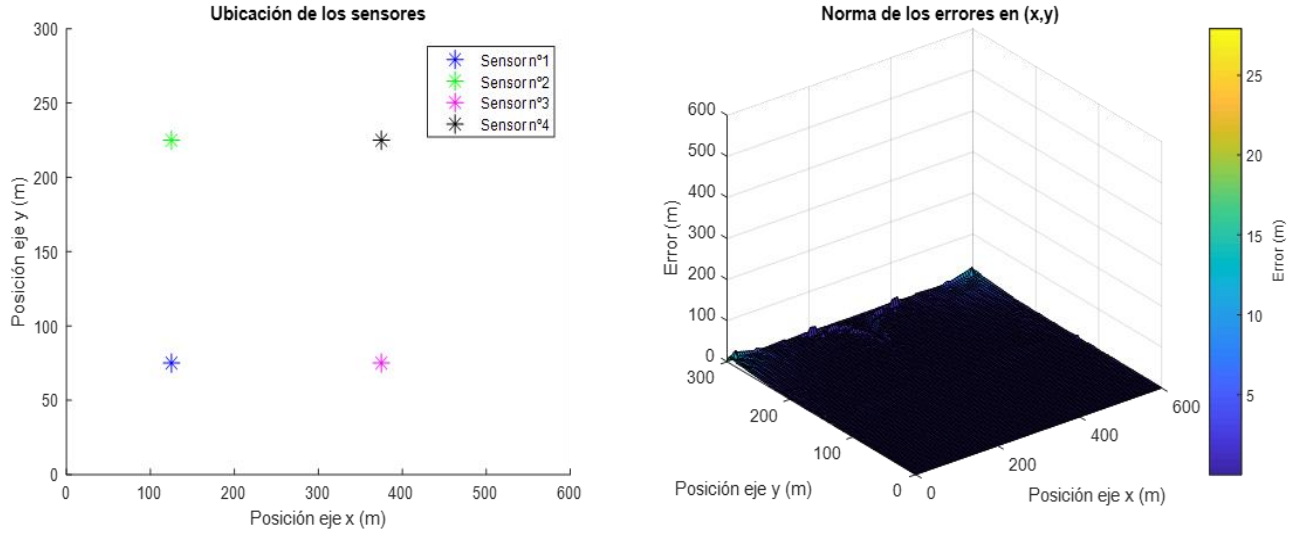

Figura 5-1 Área acotada por el caso 3 (4 sensores, 16 medidas) con resultados exactos.

Respecto al resto de objetivos, comentamos lo siguiente:

- Sobre el análisis del modelo de predicción mediante el teorema la función inversa, se ha podido completar tal y como mostramos en la sección 3.4.1 Escenarios en $\mathbb{R}^3$, demostrando que podemos obtener soluciones bien planteadas y que existen puntos no permitidos, para los que el determinante de la matriz jacobiana no es igual a cero. Se valora con un porcentaje de cumplimiento del 100%.
- En cuanto a la comprobación del programa desarrollado en escenarios académicos para su validación, se han realizado una serie de mallados que demuestran resultados favorables, permitiendo acotar el área en el que obtenemos soluciones exactas. Sin embargo, no podemos analizar escenarios más realistas, como pueden ser los escenarios urbanos, ya que las hipótesis consideradas para la solución particular *Gaussian Puff* no serían válidas. Por ejemplo, no podríamos considerar la velocidad del viento constante en el eje x. Por tanto, este objetivo se valora con un porcentaje de cumplimiento del 90%.
- Respecto al área que los sensores pueden abarcar para obtener una solución válida, hemos obtenido mediante el mallado de la sección 4.3.4 Cuarto mallado, una superficie de $600\ m \times 300\ m$. Aunque el análisis de impacto de la posición de la fuente de emisión sea un parámetro principal, también podríamos analizar otros, como el punto de partida del algoritmo x_0 mediante otros mallados. En base a ello, se valora el objetivo con un porcentaje de cumplimiento del 90%.

En definitiva, mediante este trabajo, hemos iniciado los pasos a una necesidad de la Dirección General de Armamento y Material del Ministerio de Defensa del Estado Español, descrita en la Estrategia de Tecnología e Innovación para la Defensa (ETID-2015) [4]. También es una propuesta académica en el ámbito de las matemáticas aplicadas [11] y una pequeña contribución a la Organización para la Prohibición de las Armas Químicas (OPAQ) [3] en su esfuerzo por la prohibición y eliminación de las armas químicas.

5.3 Líneas futuras

En base a lo enunciado respecto a los objetivos cumplidos, podemos abrir las siguientes líneas futuras relacionadas:

- Dada las limitaciones del modelo descrito en la publicación [10] para escenarios urbanos, es necesario emplear modelos más sofisticados basados en la resolución numérica de

ecuaciones en derivadas parciales. En la publicación [10] se propone el empleo de dichos modelos, bajo el epígrafe *Using Diffusion Models.*

- El impacto de la posición inicial del algoritmo x_0 resulta interesante de analizar mediante mallados, de manera similar a los realizados en la sección 4.3 Análisis de impacto de la posición de la fuente.
- Por último, también se propone la realización de una interfaz gráfica que implemente el programa desarrollado y que permita monitorizar la amenaza. Dicho algoritmo también podría ser integrado en software específicos que se emplean por los distintos organismos para analizar incidentes NRBQ.

6 BIBLIOGRAFÍA

[1] L.Szinicz, «History of chemical and biological warfare agents,» *Elsevier*, vol. I, nº 214, pp. 167-181, 2005.

[2] S. Chauhan, R. D'Cruz, S. Faruqi, K. Singh, S.Varma, M.Singh y V. Karthik, «Environmental Toxicology and Pharmacology,» *Elsevier*, vol. I, nº 26, pp. 113-122, 2008.

[3] Organisation for the Prohibition of Chemical Weapons, «Organisation for the Prohibition of Chemical Weapons,» [En línea]. Available: https://www.opcw.org/. [Último acceso: 26 02 2018].

[4] Gobierno de España; Ministerio de Defensa; Secretaría de Estado de Defensa; Dirección General de Armamento y Material, «Estrategia de Tecnología e Innovación para la Defensa (ETID 2015),» 2015.

[5] Gobierno de España, Ministerio de Defensa, Secretaría de Estado de Defensa, Dirección General de Armamento y Material, «Portal de Tecnología e Innovación del Ministerio de Defensa,» [En línea]. Available: http://www.tecnologiaeinnovacion.defensa.gob.es/es-es/Contenido/Paginas/detallepublicacion.aspx?publicacionID=205. [Último acceso: 26 02 2018].

[6] NATO Standardization Office (NSO), D-Paper on Chemical Downwind Hazard Distance (DHD) Calculation Methodology, 2017.

[7] R. C. Gupta, Handbook of Toxicology of Chemical Warfare Agents, Elsevier.

[8] US Army, «National Museum of Health and Medicine,» [En línea]. Available: http://www.medicalmuseum.mil/. [Último acceso: 18 02 2018].

[9] «Armas de guerra y conflictos armados,» [En línea]. Available: https://aquellasarmasdeguerra.wordpress.com/2015/03/09/algunas-armas-utilizadas-en-la-guerra-iran-irak-1980-1988-parte-1/. [Último acceso: 03 02 2018].

[10] NATO Standardization Office (NSO), ATP-45(F) Warning and Reporting and Hazard Prediction of Chemical, Biological, Radiological and Nuclear Incidents (Operators Manual), 2017.

[11] 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.

[12] J. E.Marsden y A. J. Tromba, Cálculo Vectorial, New York: Pearson, 1996.

[13] «MathWorks,» [En línea]. Available: https://es.mathworks.com/. [Último acceso: 27 Enero 2018].

[14] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/symbolic/jacobian.html?s_tid=srchtitle. [Último acceso: 02 03 2017].

[15] MathWorks, [En línea]. Available: https://es.mathworks.com/help/matlab/ref/det.html?searchHighlight=det&s_tid=doc_srchtitle. [Último acceso: 03 02 2018].

[16] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/symbolic/solve.html?searchHighlight=solve&s_tid=doc_srchtitle. [Último acceso: 03 02 2018].

[17] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/matlab/ref/global.html?searchHighlight=global&s_tid=doc_srchtitle. [Último acceso: 06 02 2018].

[18] United States Environmental Protection Agency, «Environmental Protection Agency,» [En línea]. Available: https://www.epa.gov/aegl/access-acute-exposure-guideline-levels-aegls-values#chemicals. [Último acceso: 06 02 2017].

[19] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/simulink/slref/if.html?searchHighlight=if&s_tid=doc_srchtitle. [Último acceso: 06 02 2018].

[20] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/symbolic/subs.html?searchHighlight=subs&s_tid=doc_srchtitle. [Último acceso: 06 02 2018].

[21] J. d. B. Román, Cálculo infinitesimal de una variable, Madrid: McGraw-Hill, 2004.

[22] MathWorks, «MathWorks,» [En línea]. Available: https://es.mathworks.com/help/optim/ug/lsqnonlin.html?searchHighlight=lsqnonlin&s_tid=doc_srchtitle. [Último acceso: 19 02 2018].

[23] O. Seland, A. Van Pul, A. Sorteberg y J.-P. Tuovinen, Implementation of a Resistance Dry Deposition module and a Variable Local Correction Factor in the Langrangian EMAP model, Oslo: Norwegian Meteorological Institute, 1995.

[24] EPA (United States Environmental Protection Agency), «Acute Exposure Guideline Levels for Selected Airborne Chemicals,» The National Academies Press, Washington, D.C., 2001.

[25] EPA (United States Environmental Protection Agency), «EPA (United States Environmental Protection Agency),» [En línea]. Available: https://www.epa.gov/aegl/agent-vx-results-aegl-program. [Último acceso: 22 02 2018].

ANEXO I: CÓDIGO FUENTE

AI.1 funcion_inversa.m

```
% *************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: funcion_inversa
% Descripción: pequeño script para comprobar las condiciones del teorema
% de la función inversa en nuestro problema de caracterización del término
% fuente en un incidente químico.
% *************************************************************************

% Limpiamos el workspace
clear all;
format long e;

% Selector de caso
icaso=3; % Caso de estudio

% ***********************************************************************
% Definimos las variables asociadas al problema
% ***********************************************************************

% Variables generales del problema
syms wdep;
syms gamma1;
syms gamma2;
syms A;
syms u;
syms Qs;

% Variables simbolicas asociadas al término fuente
syms xs; % Coordenada x del termino fuente
syms ys; % Coordenada y del termino fuente
syms zs; % Coordenada z del termino fuente

% Variables generales para evaluar el modelo de dispersión
syms x;
syms y;
syms z;
syms t;

% Variables simbolicas asociadas a las coordenadas de medición
syms xM1;
syms yM1;
syms zM1;
syms xM2;
syms yM2;
syms zM2;
syms xM3;
syms yM3;
syms zM3;

% Variables simbolicas asociadas a los tiempos de medicion
syms tM1;
syms tM2;
syms tM3;
```

```
% Variables simbólicas asociadas a las mediciones realizadas

% Caso 1: un punto dos tiempos y un punto un tiempo
syms DMp1t1;
syms DMp1t2;
syms DMp2t1;

% Caso 2: 3 puntos y un tiempo
syms DMp1t1;
syms DMp2t1;
syms DMp3t1;

% Caso 3: 1 punto en tres tiempos
syms DMp1t1;
syms DMp1t2;
syms DMp1t3;


% ********************************************************************
% Definimos las funciones que tenemos que derivar para establecer
% las condiciones de existencia asociadas al teorema de la función
% inversa.
% ********************************************************************

% Calculamos los valores de sigmax, sigmay y sigmaz
sigmax=1;
sigmay=1;
sigmaz=1;

% Definimos la funcion asociada al modelo de dispersión
c=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2))*...
    exp(-(x-xs-u*t)^2/(2*sigmax^2))*...
    exp(-(y-ys)^2/(2*sigmay^2))*...
    (exp(-(z-zs)^2/(2*sigmaz^2))+exp(-(z+zs)^2/(2*sigmaz^2)));

% Inicializamos la función de varias variables
F=sym(zeros(3,1));

% Caso 1: dos puntos y dos tiempos
% --------------------------------

if (icaso==1)

    % Imprimimos por pantalla el valor del caso
    disp('Caso de estudio 1')

    % Concentración en el punto 1
    cp1=subs(c,[x,y,z],[xM1,yM1,zM1]);

    % Concentración en el punto 2
    cp2=subs(c,[x,y,z],[xM2,yM2,zM2]);

    % Primera componente del campo vectorial
    %F(1)=A*wdep*int(cp1,t,0,tM1)-DMp1t1;
    F(1)=A*wdep*subs(cp1,t,tM1)*tM1-DMp1t1;

    % Segunda componente del campo vectorial
```

```
    %F(2)=A*wdep*int(cp1,t,0,tM2)-DMp1t2;
    F(2)=A*wdep*subs(cp1,t,tM2)*tM2-DMp1t2;

    % Tercera componente del campo vectorial
    %F(3)=A*wdep*int(cp2,t,0,tM1)-DMp2t1;
    F(3)=A*wdep*subs(cp2,t,tM1)*tM1-DMp2t1;

    % Calculamos el Jacobiano del campo vectorial
    disp('Calculamos el Jacobiano');
    JF1=jacobian(F,[xs,ys,zs]);

    % Simplificamos el Jacobiano para calcular el determinante
    disp('Simplificamos el Jacobiano');
    JF1=simplify(JF1);
    JF1=((sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2))/(Qs*A*wdep))*JF1;

    % Calculamos ahora el determinante del jacobiano
    disp('Calculamos el determinante');
    detJF1=det(JF1);

    % Imprimimos el valor del determinante
    disp('Valor del determinante');
    disp(detJF1);

    % Calculamos los puntos que hacen que el determinante sea cero
    PM2=solve(detJF1,[xM2,yM2,zM2]);
    PM2.yM2=simplify(PM2.yM2);

    % Mostramos los resultados por pantalla
    disp('Punto no permitido');
    disp(PM2.xM2);
    disp(PM2.yM2);
    disp(PM2.zM2);
    disp('Segunda componente');
    pretty(PM2.yM2)

end

% Caso 2: 3 puntos y un tiempo
% --------------------------------
if (icaso==2)

    % Imprimimos por pantalla el valor del caso
    disp('Caso de estudio 2')

    % Concentración en el punto 1
    cp1=subs(c,[x,y,z],[xM1,yM1,zM1]);

    % Concentración en el punto 2
    cp2=subs(c,[x,y,z],[xM2,yM2,zM2]);

    % Concentración en el punto 3
    cp3=subs(c,[x,y,z],[xM3,yM3,zM3]);

    % Primera componente del campo vectorial
    %F(1)=A*wdep*int(cp1,t,0,tM1)-DMp1t1;
    F(1)=A*wdep*subs(cp1,t,tM1)*tM1-DMp1t1;
```

```
    % Segunda componente del campo vectorial
    %F(2)=A*wdep*int(cp2,t,0,tM1)-DMp2t1;
    F(2)=A*wdep*subs(cp2,t,tM1)*tM1-DMp2t1;

    % Tercera componente del campo vectorial
    %F(3)=A*wdep*int(cp3,t,0,tM1)-DMp3t1;
    F(3)=A*wdep*subs(cp3,t,tM1)*tM1-DMp3t1;

    % Calculamos el Jacobiano del campo vectorial
    disp('Calculamos el Jacobiano');
    JF2=jacobian(F,[xs,ys,zs]);

    % Simplificamos el Jacobiano
    disp('Simplificamos el Jacobiano');
    JF2=simplify(JF2);
    JF2=(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2))*JF2/(Qs*A*wdep);

    % Calculamos ahora el determinante del jacobiano
    disp('Calculamos el determinante');
    detJF2=det(JF2);

    % Simplificamos el valor del determinante
    disp('Simplificamos el valor del determinante');
    detJF2=simplify(detJF2);

    % Imprimimos el valor del determinante
    disp('Valor del determinante');
    disp(detJF2);

    % Calculamos los puntos que hacen que el determinante sea cero
    PM=solve(detJF2,[xM3,yM3,zM3]);

    % Mostramos los resultados por pantalla
    disp('Punto no permitido');
    disp(PM.xM3);
    disp(PM.yM3);
    disp(PM.zM3);


end

% Caso 3: 1 punto y cuatro tiempos
% --------------------------------

if (icaso==3)

    % Imprimimos por pantalla el valor del caso
    disp('Caso de estudio 3')

    % Concentración en el punto 1
    cp1=subs(c,[x,y,z],[xM1,yM1,zM1]);

    % Primera componente del campo vectorial
    %F(1)=A*wdep*int(cp1,t,0,tM1)-DMp1t1;
    F(1)=A*wdep*subs(cp1,t,tM1)*tM1-DMp1t1;

    % Segunda componente del campo vectorial
    %F(2)=A*wdep*int(cp1,t,0,tM2)-DMp1t2;
```

```
    F(2)=A*wdep*subs(cp1,t,tM2)*tM2-DMp1t2;

    % Tercera componente del campo vectorial
    %F(3)=A*wdep*int(cp1,t,0,tM3)-DMp1t3;
    F(3)=A*wdep*subs(cp1,t,tM3)*tM3-DMp1t3;

    % Calculamos el Jacobiano del campo vectorial
    disp('Calculamos el Jacobiano');
    JF3=jacobian(F,[xs,ys,zs]);

    % Simplificamos el Jacobiano
    disp('Simplificamos el Jacobiano');
    JF3=simplify(JF3);
    JF3=(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2))*JF3/(Qs*A*wdep);

    % Calculamos ahora el determinante del jacobiano
    disp('Calculamos el determinante');
    detJF3=det(JF3);

    % Imprimimos el valor del determinante
    disp('Valor del determinante');
    disp(detJF3);

end
```

AI.2 datos.m

```
% ************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: datos
% Descripción: función en donde definimos los datos asociados al problema
% ************************************************************************

function datos()

    % Declaracion de las variables globales
    global icaso; % Caso de estudio
    global u; % Velocidad del viento
    global Qs; % Término fuente
    global A; % Diámetro del colector
    global wdep; % Velocidad de deposición
    global xM1; % Posición x del primer sensor
    global yM1; % Posición y del primer sensor
    global zM1; % Posición z del primer sensor
    global xM2; % Posisión x del segundo sensor
    global yM2; % Posición y del segundo sensor
    global zM2; % Posición z del segundo sensor
    global xM3; % Posisión x del tercer sensor
    global yM3; % Posición y del tercer sensor
    global zM3; % Posición z del tercer sensor
    global xM4; % Posisión x del cuarto sensor
    global yM4; % Posición y del cuarto sensor
    global zM4; % Posición z del cuarto sensor
    global tM1; % Tiempo de medida 1
    global tM2; % Tiempo de medida 2
    global tM3; % Tiempo de medida 3
    global tM4; % Tiempo de medida 4
```

```
global xs; % Posición x para generar las medidas sintéticas
global ys; % Posición y para generar las medidas sintéticas
global zs; % Posición z para generar las medidas sintéticas
global S; % Estabilidad atmosférica
global DHD; % Distancia de riesgo a sotavento

% Establecemos los datos asociados al caso de estudio
icaso=2;
A=0.16; % m2
u=1.02889; % m/s ////(2 kts) tabla AEP-45
Qs=4.0332; % kg ////4 litros de VX(dVX=1008.3 kg/m3) tabla AEP-45
S=1; % estabilidad atmosférica tabla AEP-45
DHD=1300; % m //////1.3 km tabla AEP-45
wdep=vd(u,S); % Calculamos la velocidad de deposición

% Fijamos las coordenadas espacio-temporales de los puntos de medida
if(icaso==1) % Caso 1
    xM1=100;
    yM1=200;
    zM1=1.0;

    xM2=200;
    yM2=400;
    zM2=3.0;

    tM1=160.0;
    tM2=200.0;

end

if(icaso==2) % Caso 2
    xM1=100;
    yM1=200;
    zM1=1.0;

    xM2=200;
    yM2=400;
    zM2=3.0;

    xM3=300;
    yM3=250;
    zM3=5.0;

    tM1=160;

end

if(icaso==3) % Caso 3

    xM1=125;
    yM1=75;
    zM1=1;

    xM2=125;
    yM2=225;
    zM2=2;

    xM3=375;
```

```
        yM3=75;
        zM3=3;

        xM4=375;
        yM4=225;
        zM4=4;

        tM1=100;
        tM2=200;
        tM3=300;
        tM4=400;

    end

    % Fijamos el punto de vertido para generar las medidas sintéticas
    xs=150;
    ys=200;
    zs=2;

end
```

AI.3 medidas.m

```
% ********************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: medidas
% Descripción: función de Matlab que genera las medidas sintéticas
% ********************************************************************

function medidas()

    % Declaracion de las variables globales
    global icaso; % Caso de estudio
    global sigmax; % Coeficiente asociado al termino de difusion
    global sigmay; % Coeficiente asociado al termino de difusion
    global sigmaz; % Coeficiente asociado al termino de difusion
    global u; % Velocidad del viento
    global Qs; % Término fuente
    global A; % Diámetro del colector
    global wdep; % Velocidad de deposición
    global xM1; % Posición x del primer sensor
    global yM1; % Posición y del primer sensor
    global zM1; % Posición z del primer sensor
    global xM2; % Posisión x del segundo sensor
    global yM2; % Posición y del segundo sensor
    global zM2; % Posición z del segundo sensor
    global xM3; % Posisión x del tercer sensor
    global yM3; % Posición y del tercer sensor
    global zM3; % Posición z del tercer sensor
    global xM4; % Posisión x del cuarto sensor
    global yM4; % Posición y del cuarto sensor
    global zM4; % Posición z del cuarto sensor
    global tM1; % Tiempo de medida 1
    global tM2; % Tiempo de medida 2
    global tM3; % Tiempo de medida 3
    global tM4; % Tiempo de medida 4
    global DMp1t1; % Medida sensor 1 en tiempo 1
```

```
global DMp2t1; % Medida sensor 2 en tiempo 1
global DMp3t1; % Medida sensor 3 en tiempo 1
global DMp4t1; % Medida sensor 4 en tiempo 1
global DMp1t2; % Medida sendor 1 en tiempo 2
global DMp2t2; % Medida sendor 2 en tiempo 2
global DMp3t2; % Medida sendor 3 en tiempo 2
global DMp4t2; % Medida sendor 4 en tiempo 2
global DMp1t3; % Medida sendor 1 en tiempo 3
global DMp2t3; % Medida sendor 2 en tiempo 3
global DMp3t3; % Medida sendor 3 en tiempo 3
global DMp4t3; % Medida sendor 4 en tiempo 3
global DMp1t4; % Medida sendor 1 en tiempo 4
global DMp2t4; % Medida sendor 2 en tiempo 4
global DMp3t4; % Medida sendor 3 en tiempo 4
global DMp4t4; % Medida sendor 4 en tiempo 4
global xs; % Posición x para generar las medidas sintéticas
global ys; % Posición y para generar las medidas sintéticas
global zs; % Posición z para generar las medidas sintéticas

% Definimos una serie de variables simbólicas necesarias
syms x;
syms y;
syms z;
syms t;

% Definimos el término constante
ctemp=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2));

% Definimos la función de concentración
c=ctemp*exp(-(x-xs-u*t)^2/(2*sigmax^2))*...
        exp(-(y-ys)^2/(2*sigmay^2))*...
       (exp(-(z-zs)^2/(2*sigmaz^2))+exp(-(z+zs)^2/(2*sigmaz^2)));

% Generamos las medidas correspondientes al caso 1
if(icaso==1)
    % Calculamos la concentración en el punto PM1 y tiempo tM1
    cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
    % Calculamos la concentración en el punto PM1 y tiempo tM2
    cp1t2=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM2]);
    % Calculamos la concentración en el punto PM2 y tiempo tM1
    cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
    % Calculamos la medición asociada a (PM1,tM1)
    DMp1t1=A*wdep*cp1t1*tM1;
    % Calculamos la medición asociada a (PM1,tM2)
    DMp1t2=A*wdep*cp1t2*tM2;
    % Calculamos la medición asociada a (PM2,tM1)
    DMp2t1=A*wdep*cp2t1*tM1;
end

% Generamos las medidas correspondientes al caso 2
if(icaso==2)
    % Calculamos la concentración en el punto PM1 y tiempo tM1
    cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
    % Calculamos la concentración en el punto PM2 y tiempo tM1
    cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
    % Calculamos la concentración en el punto PM3 y tiempo tM1
    cp3t1=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM1]);
    % Calculamos la medición asociada a (PM1,tM1)
```

```
        DMp1t1=A*wdep*cp1t1*tM1;
        % Calculamos la medición asociada a (PM2,tM1)
        DMp2t1=A*wdep*cp2t1*tM1;
        % Calculamos la medición asociada a (PM3,tM1)
        DMp3t1=A*wdep*cp3t1*tM1;
    end

    % Generamos las medidas correspondientes al caso 3
    if(icaso==3)
        % Calculamos la concentración en el punto PM1 y tiempo tM1
        cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la concentración en el punto PM2 y tiempo tM1
        cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la concentración en el punto PM3 y tiempo tM1
        cp3t1=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la concentración en el punto PM4 y tiempo tM1
        cp4t1=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM1]);

        % Calculamos la concentración en el punto PM1 y tiempo tM2
        cp1t2=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM2]);
        % Calculamos la concentración en el punto PM2 y tiempo tM2
        cp2t2=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM2]);
        % Calculamos la concentración en el punto PM3 y tiempo tM2
        cp3t2=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM2]);
        % Calculamos la concentracion en el punto PM4 y tiempo tM2
        cp4t2=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM2]);

        % Calculamos la concentración en el punto PM1 y tiempo tM3
        cp1t3=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM3]);
        % Calculamos la concentración en el punto PM2 y tiempo tM3
        cp2t3=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM3]);
        % Calculamos la concentración en el punto PM3 y tiempo tM3
        cp3t3=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM3]);
        % Calculamos la concentración en el punto PM4 y tiempo tM3
        cp4t3=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM3]);

        % Calculamos la concentración en el punto PM1 y tiempo tM4
        cp1t4=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM4]);
        % Calculamos la concentración en el punto PM2 y tiempo tM4
        cp2t4=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM4]);
        % Calculamos la concentración en el punto PM3 y tiempo tM4
        cp3t4=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM4]);
        % Calculamos la concentración en el punto PM4 y tiempo tM4
        cp4t4=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM4]);

        % Calculamos la medición asociada a (PM1,tM1)
        DMp1t1=A*wdep*cp1t1*tM1;
        % Calculamos la medición asociada a (PM2,tM1)
        DMp2t1=A*wdep*cp2t1*tM1;
        % Calculamos la medición asociada a (PM3,tM1)
        DMp3t1=A*wdep*cp3t1*tM1;
        % Calculamos la medición asociada a (PM4,tM1)
        DMp4t1=A*wdep*cp4t1*tM1;

        % Calculamos la medición asociada a (PM1,tM2)
        DMp1t2=A*wdep*cp1t2*tM2;
        % Calculamos la medición asociada a (PM2,tM2)
        DMp2t2=A*wdep*cp2t2*tM2;
```

```
        % Calculamos la medición asociada a (PM3,tM2)
        DMp3t2=A*wdep*cp3t2*tM2;
        % Calculamos la medición asociada a (PM4,tM2)
        DMp4t2=A*wdep*cp4t2*tM2;

        % Calculamos la medición asociada a (PM1,tM3)
        DMp1t3=A*wdep*cp1t3*tM3;
        % Calculamos la medición asociada a (PM2,tM3)
        DMp2t3=A*wdep*cp2t3*tM3;
        % Calculamos la medición asociada a (PM3,tM3)
        DMp3t3=A*wdep*cp3t3*tM3;
        % Calculamos la medición asociada a (PM4,tM3)
        DMp4t3=A*wdep*cp4t3*tM3;

        % Calculamos la medición asociada a (PM1,tM4)
        DMp1t4=A*wdep*cp1t4*tM4;
        % Calculamos la medición asociada a (PM2,tM4)
        DMp2t4=A*wdep*cp2t4*tM4;
        % Calculamos la medición asociada a (PM3,tM4)
        DMp3t4=A*wdep*cp3t4*tM4;
        % Calculamos la medición asociada a (PM4,tM4)
        DMp4t4=A*wdep*cp4t4*tM4;

    end
end
```

AI.4 sigma.m

```
%*************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: sigma
% Descripción: pequeño script para calcular el valor de sigma
%*************************************************************

function sigma()

  % Declaración de las variables globales
  global sigmax;
  global sigmay;
  global sigmaz;
  global xs;
  global S;
  global DHD;

  % Definimos la variable simbólica x
  syms x;

  % Definimos los parámetros asociados a la función de sigma
  F=1.041+0.445*S-9.359e-2*S.^2+6.322e-3*S.^3;
  f=0.783-5.47e-2*S+1.084e-2*S.^2-7.016e-4*S.^3;

  G=0.1229*exp(0.3295*S);
  g=0.97-0.09*S;

  % Definimos la función de sigma
  fsigmax=F*abs(x-xs).^f;
  fsigmay=F*abs(x-xs).^f;
```

```
  fsigmaz=G*abs(x-xs).^g;

  % Calculamos el valor medio de sigma
  sigmax=(int(fsigmax,x,xs,DHD))/DHD;
  sigmay=(int(fsigmay,x,xs,DHD))/DHD;
  sigmaz=(int(fsigmaz,x,xs,DHD))/DHD;

end
```

AI.5 SolveProblem.m

```
%**************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: SolveProblem
% Descripción: pequeño script para lanzar la resolución del problema
%**************************************************************************

% Limpiamos el workspace
clear global
clear all
format long e;

% Declaramos las variables globales que emplearemos
global DHD;
global icaso;

% Llamamos a la función de datos
datos;

% Generamos los valores de sigma
sigma;

% Generamos las medidas
medidas;

% Opciones para el algoritmo de optimización
options=optimset('Jacobian','on','Display','iter','MaxIter',50,...
    'TolFun',1.e-21,'MaxFunEvals',1000);

% Punto inicial
x0=[325;325;2.5];

% Cotas inferiores para el punto de vertido
lb=[0;0;0];

% Cotas superiores para el punto de vertido
ub=[DHD;DHD;5];

% Lanzamos el algoritmo
[xsol]=lsqnonlin(@EvalCoste,x0,lb,ub,options);

% Mostramos información del resultado
disp('Solución');
disp(xsol);
```

AI.6 vd.m

```
% ************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: vd
% Descripción: función que calcula la velocidad de deposición
% ************************************************************************

function z=vd(wind_mps,stab)

    % Declaramos el valor de z0 (roughness length)
    z0=0.2; % z0=0.2 para tierra y 0.02 para mar

    % Declaramos un parámetro que se empleará para hacer los cálculos
    kappa=0.41;

    % Calculamos el valor de a (parameter describing the vegetation)
    a=max(100,500-400*(z0-0.01)/2);

    % Calculamos el valor de u*
    ustar=wind_mps*kappa/log((10.+z0)/z0);

    % Analizamos los casos en función de la estabilidad
    if(stab>3.5)

        % Calculamos el valor de vd
        z=ustar/a;
    else
        % Calculamos el valor de L (Monin-Obukhov Length)
        xobu=1./(-0.04+(stab-2.5)*(0.08)/3.);

        % Calculamos el valor final de vd
        z=(1-300/xobu)^(2/3)*ustar/a;
    end

end
```

AI.7 fun.m

```
% ********************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: fun
% Descripción: función asociada al problema de mínimos cuadrados que
% calcula el error cuadrático en cada uno de los puntos de medición
%*********************************************************************

function F=fun(p)

    % Declaración de las variables globales
    global icaso; % Caso de estudio
    global sigmax; % Coeficiente asociado al término de difusión
    global sigmay; % Coeficiente asociado al término de difusión
    global sigmaz; % Coeficiente asociado al término de difusión
    global u; % Velocidad del viento
    global Qs; % Término fuente
    global A; % Diámetro del colector
```

```
global wdep; % Velocidad de deposición
global xM1; % Posición x del primer sensor
global yM1; % Posición y del primer sensor
global zM1; % Posición z del primer sensor
global xM2; % Posisión x del segundo sensor
global yM2; % Posición y del segundo sensor
global zM2; % Posición z del segundo sensor
global xM3; % Posisión x del tercer sensor
global yM3; % Posición y del tercer sensor
global zM3; % Posición z del tercer sensor
global xM4; % Posisión x del cuarto sensor
global yM4; % Posición y del cuarto sensor
global zM4; % Posición z del cuarto sensor
global tM1; % Tiempo de medida 1
global tM2; % Tiempo de medida 2
global tM3; % Tiempo de medida 3
global tM4; % Tiempo de medida 4
global DMp1t1; % Medida sensor 1 en tiempo 1
global DMp2t1; % Medida sensor 2 en tiempo 1
global DMp3t1; % Medida sensor 3 en tiempo 1
global DMp4t1; % Medida sensor 4 en tiempo 1
global DMp1t2; % Medida sendor 1 en tiempo 2
global DMp2t2; % Medida sendor 2 en tiempo 2
global DMp3t2; % Medida sendor 3 en tiempo 2
global DMp4t2; % Medida sendor 4 en tiempo 2
global DMp1t3; % Medida sendor 1 en tiempo 3
global DMp2t3; % Medida sendor 2 en tiempo 3
global DMp3t3; % Medida sendor 3 en tiempo 3
global DMp4t3; % Medida sendor 4 en tiempo 3
global DMp1t4; % Medida sendor 1 en tiempo 4
global DMp2t4; % Medida sendor 2 en tiempo 4
global DMp3t4; % Medida sendor 3 en tiempo 4
global DMp4t4; % Medida sendor 4 en tiempo 4

% Declaramos la variable de salida
if((icaso==1)||(icaso==2))
    F=zeros(3,1);
end
if(icaso==3)
    F=zeros(16,1);
end

% Extraemos el punto de vertido de la variable x
xsp=p(1);
ysp=p(2);
zsp=p(3);

% Definimos una serie de variables simbólicas necesarias
syms x;
syms y;
syms z;
syms t;

% Definimos el término constante
ctemp=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2));

% Definimos la función de concentración
c=ctemp*exp(-(x-xsp-u*t)^2/(2*sigmax^2))*...
```

```
            exp(-(y-ysp)^2/(2*sigmay^2))*...
           (exp(-(z-zsp)^2/(2*sigmaz^2))+exp(-(z+zsp)^2/(2*sigmaz^2)));

    % Calculamos la función F en el caso 1
    if(icaso==1)
        % Calculamos la concentración en el punto PM1 y tiempo tM1
        cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la concentración en el punto PM1 y tiempo tM2
        cp1t2=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM2]);
        % Calculamos la concentración en el punto PM2 y tiempo tM1
        cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la primera componente de F
        F(1)=(A*wdep*cp1t1*tM1-DMp1t1);
        % Calculamos la segunda componente de F
        F(2)=(A*wdep*cp1t2*tM2-DMp1t2);
        % Calculamos la tercera componente de F
        F(3)=(A*wdep*cp2t1*tM1-DMp2t1);

    end

    % Calculamos la función F en el caso 2
    if(icaso==2)
        % Calculamos la concentración en el punto PM1 y tiempo tM1
        cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la concentración en el punto PM2 y tiempo tM1
        cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la concentración en el punto PM3 y tiempo tM1
        cp3t1=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la primera componente de F
        F(1)=(A*wdep*cp1t1*tM1-DMp1t1);
        % Calculamos la segunda componente de F
        F(2)=(A*wdep*cp2t1*tM1-DMp2t1);
        % Calculamos la tercera componente de F
        F(3)=(A*wdep*cp3t1*tM1-DMp3t1);
    end

    % Calculamos la función F en el caso 3
    if(icaso==3)
        % Calculamos la concentración en el punto PM1 y tiempo tM1
        cp1t1=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la concentración en el punto PM2 y tiempo tM1
        cp2t1=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la concentración en el punto PM3 y tiempo tM1
        cp3t1=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la concentración en el punto PM4 y tiempo tM1
        cp4t1=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM1]);

        % Calculamos la concentración en el punto PM1 y tiempo tM2
        cp1t2=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM2]);
        % Calculamos la concentración en el punto PM2 y tiempo tM2
        cp2t2=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM2]);
        % Calculamos la concentración en el punto PM3 y tiempo tM2
        cp3t2=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM2]);
        % Calculamos la concentración en el punto PM4 y tiempo tM2
        cp4t2=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM2]);

        % Calculamos la concentración en el punto PM1 y tiempo tM3
        cp1t3=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM3]);
```

```
        % Calculamos la concentración en el punto PM2 y tiempo tM3
        cp2t3=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM3]);
        % Calculamos la concentración en el punto PM3 y tiempo tM3
        cp3t3=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM3]);
        % Calculamos la concentración en el punto PM4 y tiempo tM3
        cp4t3=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM3]);

        % Calculamos la concentración en el punto PM1 y tiempo tM4
        cp1t4=subs(c,[x,y,z,t],[xM1,yM1,zM1,tM4]);
        % Calculamos la concentración en el punto PM2 y tiempo tM4
        cp2t4=subs(c,[x,y,z,t],[xM2,yM2,zM2,tM4]);
        % Calculamos la concentración en el punto PM3 y tiempo tM4
        cp3t4=subs(c,[x,y,z,t],[xM3,yM3,zM3,tM4]);
        % Calculamos la concentración en el punto PM4 y tiempo tM4
        cp4t4=subs(c,[x,y,z,t],[xM4,yM4,zM4,tM4]);

        % Calculamos la primera componente de F
        F(1)=(A*wdep*cp1t1*tM1-DMp1t1);
        % Calculamos la segunda componente de F
        F(2)=(A*wdep*cp2t1*tM1-DMp2t1);
        % Calculamos la tercera componente de F
        F(3)=(A*wdep*cp3t1*tM1-DMp3t1);
        % Calculamos la cuarta componente de F
        F(4)=(A*wdep*cp4t1*tM1-DMp4t1);

        % Calculamos la primera componente de F
        F(5)=(A*wdep*cp1t2*tM2-DMp1t2);
        % Calculamos la segunda componente de F
        F(6)=(A*wdep*cp2t2*tM2-DMp2t2);
        % Calculamos la tercera componente de F
        F(7)=(A*wdep*cp3t2*tM2-DMp3t2);
        % Calculamos la cuarta componente de F
        F(8)=(A*wdep*cp4t2*tM2-DMp4t2);

        % Calculamos la primera componente de F
        F(9)=(A*wdep*cp1t3*tM3-DMp1t3);
        % Calculamos la segunda componente de F
        F(10)=(A*wdep*cp2t3*tM3-DMp2t3);
        % Calculamos la tercera componente de F
        F(11)=(A*wdep*cp3t3*tM3-DMp3t3);
        % Calculamos la cuarta componente de F
        F(12)=(A*wdep*cp4t3*tM3-DMp4t3);

        % Calculamos la primera componente de F
        F(13)=(A*wdep*cp1t4*tM4-DMp1t4);
        % Calculamos la segunda componente de F
        F(14)=(A*wdep*cp2t4*tM4-DMp2t4);
        % Calculamos la tercera componente de F
        F(15)=(A*wdep*cp3t4*tM4-DMp3t4);
        % Calculamos la cuarta componente de F
        F(16)=(A*wdep*cp4t4*tM4-DMp4t4);

    end

end
```

AI.8 dfun.m

```
% ********************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: dfun
% Descripción: función asociada al problema de mínimos cuadrados que
% calcula el error cuadrático en cada uno de los puntos de medición
%*********************************************************************

function JF=dfun(p)

    % Declaración de las variables globales
    global icaso; % Caso de estudio
    global sigmax; % Coeficiente asociado al término de difusión
    global sigmay; % Coeficiente asociado al término de difusión
    global sigmaz; % Coeficiente asociado al término de difusión
    global A; % Diámetro del colector
    global wdep; % Velocidad de deposición
    global u; % Velocidad del viento
    global Qs; % Término fuente
    global xM1; % Posición x del primer sensor
    global yM1; % Posición y del primer sensor
    global zM1; % Posición z del primer sensor
    global xM2; % Posisión x del segundo sensor
    global yM2; % Posición y del segundo sensor
    global zM2; % Posición z del segundo sensor
    global xM3; % Posisión x del tercer sensor
    global yM3; % Posición y del tercer sensor
    global zM3; % Posición z del tercer sensor
    global xM4; % Posisión x del cuarto sensor
    global yM4; % Posición y del cuarto sensor
    global zM4; % Posición z del cuarto sensor
    global tM1; % Tiempo de medida 1
    global tM2; % Tiempo de medida 2
    global tM3; % Tiempo de medida 3
    global tM4; % Tiempo de medida 4

    % Declaramos la variable de salida
    if((icaso==1)||(icaso==2))
        JF=zeros(3,3);
    end
    if(icaso==3)
        JF=zeros(16,3);
    end

    % Extraemos el punto de vertido de la variable x
    xsp=p(1);
    ysp=p(2);
    zsp=p(3);

    % Definimos una serie de variables simbólicas necesarias
    syms x;
    syms y;
    syms z;
    syms t;

    % Definimos el término constante
    ctemp=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2));
```

```
% Definimos la funcion de concentración
c=ctemp*exp(-(x-xsp-u*t)^2/(2*sigmax^2))*...
        exp(-(y-ysp)^2/(2*sigmay^2))*...
       (exp(-(z-zsp)^2/(2*sigmaz^2))+exp(-(z+zsp)^2/(2*sigmaz^2)));

% Definimos la derivada de la concentración con respecto a xsp
dxspc=(-(exp(-(y - ysp)^2/(2*sigmay^2))*exp(-(xsp - x + ...
    t*u)^2/(2*sigmax^2))*(exp(-(z + zsp)^2/(2*sigmaz^2)) + ...
    exp(-(z - zsp)^2/(2*sigmaz^2)))*(2*xsp - 2*x + ...
    2*t*u))/(2*sigmax^2))*ctemp;

% Definimos la derivada de la concentración con respecto a ysp
dyspc=((exp(-(y - ysp)^2/(2*sigmay^2))*exp(-(xsp - x + ...
    t*u)^2/(2*sigmax^2))*(2*y - 2*ysp)*(exp(-(z + ...
    zsp)^2/(2*sigmaz^2)) + exp(-(z - ...
    zsp)^2/(2*sigmaz^2))))/(2*sigmay^2))*ctemp;

% Definimos la derivada de la concentración con respecto a zsp
dzspc=(-exp(-(y - ysp)^2/(2*sigmay^2))*exp(-(xsp - x + ...
    t*u)^2/(2*sigmax^2))*((exp(-(z + ...
    zsp)^2/(2*sigmaz^2))*(2*z + 2*zsp))/(2*sigmaz^2) - ...
    (exp(-(z - zsp)^2/(2*sigmaz^2))*(2*z - ...
    2*zsp))/(2*sigmaz^2)))*ctemp;

% Calculamos el Jacobiano de la función en el caso 1
if(icaso==1)
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM1 con respecto a xsp
    dxspcp1t1=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM1 con respecto a ysp
    dyspcp1t1=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM1 con respecto a zsp
    dzspcp1t1=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM2 con respecto a xsp
    dxspcp1t2=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM2 con respecto a ysp
    dyspcp1t2=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);
    % Calculamos la derivada de la concentración en el punto
    % PM1 y tiempo tM2 con respecto a zsp
    dzspcp1t2=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);
    % Calculamos la derivada de la concentración en el punto
    % PM2 y tiempo tM1 con respecto a xsp
    dxspcp2t1=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
    % Calculamos la derivada de la concentración en el punto
    % PM2 y tiempo tM1 con respecto a ysp
    dyspcp2t1=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
    % Calculamos la derivada de la concentración en el punto
    % PM2 y tiempo tM1 con respecto a zsp
    dzspcp2t1=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
    % Calculamos la primera fila de la matriz Jacobiana
    JF(1,:)=A*wdep*tM1*[dxspcp1t1,dyspcp1t1,dzspcp1t1];
    % Calculamos la segunda fila de la matriz Jacobiana
    JF(2,:)=A*wdep*tM2*[dxspcp1t2,dyspcp1t2,dzspcp1t2];
```

```
        % Calculamos la tercera fila de la matriz Jacobiana
        JF(3,:)=A*wdep*tM1*[dxspcp2t1,dyspcp2t1,dzspcp2t1];
    end

    % Calculamos el Jacobiano de la funcion en el caso 2
    if(icaso==2)
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a xsp
        dxspcp1t1=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a ysp
        dyspcp1t1=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a zsp
        dzspcp1t1=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM2 y tiempo tM1 con respecto a xsp
        dxspcp2t1=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM2 y tiempo tM1 con respecto a ysp
        dyspcp2t1=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM2 y tiempo tM1 con respecto a zsp
        dzspcp2t1=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM3 y tiempo tM1 con respecto a xsp
        dxspcp3t1=subs(dxspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM3 y tiempo tM1 con respecto a ysp
        dyspcp3t1=subs(dyspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM3 y tiempo tM1 con respecto a zsp
        dzspcp3t1=subs(dzspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);
        % Calculamos la primera fila de la matriz Jacobiana
        JF(1,:)=A*wdep*tM1*[dxspcp1t1,dyspcp1t1,dzspcp1t1];
        % Calculamos la segunda fila de la matriz Jacobiana
        JF(2,:)=A*wdep*tM1*[dxspcp2t1,dyspcp2t1,dzspcp2t1];
        % Calculamos la tercera fila de la matriz Jacobiana
        JF(3,:)=A*wdep*tM1*[dxspcp3t1,dyspcp3t1,dzspcp3t1];

    end

    % Calculamos el Jacobiano de la funcion en el caso 3
    if(icaso==3)
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a xsp
        dxspcp1t1=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a ysp
        dyspcp1t1=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);
        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM1 con respecto a zsp
        dzspcp1t1=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM1]);

        % Calculamos la derivada de la concentración en el punto
        % PM1 y tiempo tM2 con respecto a xsp
        dxspcp1t2=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);
        % Calculamos la derivada de la concentración en el punto
```

```
% PM1 y tiempo tM2 con respecto a ysp
dyspcp1t2=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM2 con respecto a zsp
dzspcp1t2=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM2]);

% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM3 con respecto a xsp
dxspcp1t3=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM3 con respecto a ysp
dyspcp1t3=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM3 con respecto a zsp
dzspcp1t3=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM3]);

% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM4 con respecto a xsp
dxspcp1t4=subs(dxspc,[x,y,z,t],[xM1,yM1,zM1,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM4 con respecto a ysp
dyspcp1t4=subs(dyspc,[x,y,z,t],[xM1,yM1,zM1,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM1 y tiempo tM4 con respecto a zsp
dzspcp1t4=subs(dzspc,[x,y,z,t],[xM1,yM1,zM1,tM4]);

% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM1 con respecto a xsp
dxspcp2t1=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM1 con respecto a ysp
dyspcp2t1=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM1 con respecto a zsp
dzspcp2t1=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM1]);

% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM2 con respecto a xsp
dxspcp2t2=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM2 con respecto a ysp
dyspcp2t2=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM2 con respecto a zsp
dzspcp2t2=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM2]);

% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM3 con respecto a xsp
dxspcp2t3=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM3 con respecto a ysp
dyspcp2t3=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM3 con respecto a zsp
dzspcp2t3=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM3]);

% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM4 con respecto a xsp
```

```
dxspcp2t4=subs(dxspc,[x,y,z,t],[xM2,yM2,zM2,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM4 con respecto a ysp
dyspcp2t4=subs(dyspc,[x,y,z,t],[xM2,yM2,zM2,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM2 y tiempo tM4 con respecto a zsp
dzspcp2t4=subs(dzspc,[x,y,z,t],[xM2,yM2,zM2,tM4]);

% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM1 con respecto a xsp
dxspcp3t1=subs(dxspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM1 con respecto a ysp
dyspcp3t1=subs(dyspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM1 con respecto a zsp
dzspcp3t1=subs(dzspc,[x,y,z,t],[xM3,yM3,zM3,tM1]);

% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM2 con respecto a xsp
dxspcp3t2=subs(dxspc,[x,y,z,t],[xM3,yM3,zM3,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM2 con respecto a ysp
dyspcp3t2=subs(dyspc,[x,y,z,t],[xM3,yM3,zM3,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM2 con respecto a zsp
dzspcp3t2=subs(dzspc,[x,y,z,t],[xM3,yM3,zM3,tM2]);

% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM3 con respecto a xsp
dxspcp3t3=subs(dxspc,[x,y,z,t],[xM3,yM3,zM3,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM3 con respecto a ysp
dyspcp3t3=subs(dyspc,[x,y,z,t],[xM3,yM3,zM3,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM3 con respecto a zsp
dzspcp3t3=subs(dzspc,[x,y,z,t],[xM3,yM3,zM3,tM3]);

% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM4 con respecto a xsp
dxspcp3t4=subs(dxspc,[x,y,z,t],[xM3,yM3,zM3,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM4 con respecto a ysp
dyspcp3t4=subs(dyspc,[x,y,z,t],[xM3,yM3,zM3,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM3 y tiempo tM4 con respecto a zsp
dzspcp3t4=subs(dzspc,[x,y,z,t],[xM3,yM3,zM3,tM4]);

% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM1 con respecto a xsp
dxspcp4t1=subs(dxspc,[x,y,z,t],[xM4,yM4,zM4,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM1 con respecto a ysp
dyspcp4t1=subs(dyspc,[x,y,z,t],[xM4,yM4,zM4,tM1]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM1 con respecto a zsp
dzspcp4t1=subs(dzspc,[x,y,z,t],[xM4,yM4,zM4,tM1]);
```

```
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM2 con respecto a xsp
dxspcp4t2=subs(dxspc,[x,y,z,t],[xM4,yM4,zM4,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM2 con respecto a ysp
dyspcp4t2=subs(dyspc,[x,y,z,t],[xM4,yM4,zM4,tM2]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM2 con respecto a zsp
dzspcp4t2=subs(dzspc,[x,y,z,t],[xM4,yM4,zM4,tM2]);

% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM3 con respecto a xsp
dxspcp4t3=subs(dxspc,[x,y,z,t],[xM4,yM4,zM4,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM3 con respecto a ysp
dyspcp4t3=subs(dyspc,[x,y,z,t],[xM4,yM4,zM4,tM3]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM3 con respecto a zsp
dzspcp4t3=subs(dzspc,[x,y,z,t],[xM4,yM4,zM4,tM3]);

% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM4 con respecto a xsp
dxspcp4t4=subs(dxspc,[x,y,z,t],[xM4,yM4,zM4,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM4 con respecto a ysp
dyspcp4t4=subs(dyspc,[x,y,z,t],[xM4,yM4,zM4,tM4]);
% Calculamos la derivada de la concentración en el punto
% PM4 y tiempo tM4 con respecto a zsp
dzspcp4t4=subs(dzspc,[x,y,z,t],[xM4,yM4,zM4,tM4]);

% Calculamos la primera fila de la matriz Jacobiana
JF(1,:)=A*wdep*tM1*[dxspcp1t1,dyspcp1t1,dzspcp1t1];
% Calculamos la segunda fila de la matriz Jacobiana
JF(2,:)=A*wdep*tM1*[dxspcp2t1,dyspcp2t1,dzspcp2t1];
% Calculamos la tercera fila de la matriz Jacobiana
JF(3,:)=A*wdep*tM1*[dxspcp3t1,dyspcp3t1,dzspcp3t1];
% Calculamos la tercera fila de la matriz Jacobiana
JF(4,:)=A*wdep*tM1*[dxspcp4t1,dyspcp4t1,dzspcp4t1];

% Calculamos la primera fila de la matriz Jacobiana
JF(5,:)=A*wdep*tM2*[dxspcp1t2,dyspcp1t2,dzspcp1t2];
% Calculamos la segunda fila de la matriz Jacobiana
JF(6,:)=A*wdep*tM2*[dxspcp2t2,dyspcp2t2,dzspcp2t2];
% Calculamos la tercera fila de la matriz Jacobiana
JF(7,:)=A*wdep*tM2*[dxspcp3t2,dyspcp3t2,dzspcp3t2];
% Calculamos la tercera fila de la matriz Jacobiana
JF(8,:)=A*wdep*tM2*[dxspcp4t2,dyspcp4t2,dzspcp4t2];

% Calculamos la primera fila de la matriz Jacobiana
JF(9,:)=A*wdep*tM3*[dxspcp1t3,dyspcp1t3,dzspcp1t3];
% Calculamos la segunda fila de la matriz Jacobiana
JF(10,:)=A*wdep*tM3*[dxspcp2t3,dyspcp2t3,dzspcp2t3];
% Calculamos la tercera fila de la matriz Jacobiana
JF(11,:)=A*wdep*tM3*[dxspcp3t3,dyspcp3t3,dzspcp3t3];
% Calculamos la tercera fila de la matriz Jacobiana
JF(12,:)=A*wdep*tM3*[dxspcp4t3,dyspcp4t3,dzspcp4t3];

% Calculamos la primera fila de la matriz Jacobiana
```

```
        JF(13,:)=A*wdep*tM4*[dxspcp1t4,dyspcp1t4,dzspcp1t4];
        % Calculamos la segunda fila de la matriz Jacobiana
        JF(14,:)=A*wdep*tM4*[dxspcp2t4,dyspcp2t4,dzspcp2t4];
        % Calculamos la tercera fila de la matriz Jacobiana
        JF(15,:)=A*wdep*tM4*[dxspcp3t4,dyspcp3t4,dzspcp3t4];
        % Calculamos la tercera fila de la matriz Jacobiana
        JF(16,:)=A*wdep*tM4*[dxspcp4t4,dyspcp4t4,dzspcp4t4];

    end
end
```

AI.9 calculo_derivadas.m

```
% ************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: calculo_derivadas
% Descripción: pequeño script para calcular las derivadas de la
% concentración con respecto al punto de emisión
% ************************************************************

% Declaramos una serie de variables simbólicas
syms Qs;
syms x;
syms y;
syms z;
syms xsp;
syms ysp;
syms zsp;
syms gamma1;
syms gamma2;
syms u;
syms t;
syms epsilon;
syms sigmax;
syms sigmay;
syms sigmaz;

% Definimos la funcion de concentración
%c=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2))*...

c=exp(-(x-xsp-u*t)^2/(2*sigmax^2))*...
    exp(-(y-ysp)^2/(2*sigmay^2))*...
   (exp(-(z-zsp)^2/(2*sigmaz^2))+exp(-(z+zsp)^2/(2*sigmaz^2)));

% Calculamos la derivada con respecto a xsp
dxspc=diff(c,xsp)

% Calculamos la derivada con respecto a ysp
dyspc=diff(c,ysp)

% Calculamos la derivada con respecto a zsp
dzspc=diff(c,zsp)
```

AI.10 EvalCoste.m

```
% *************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: EvalCoste
% Descripción: función que evalua el funcional de coste y el jacobiano de
% las componentes
% *************************************************************************

function [F,JF]=EvalCoste(p)

  % Calculamos la función
  F=fun(p);

  % Cálculo del jacobiano si procede
  if nargout > 1
      JF=dfun(p);
  end
```

AI.11 pinta.m

```
% *************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: pinta
% Descripción: función de Matlab para pintar la evolución del penacho
% *************************************************************************

% Limpiamos el workspace
clear global
clear all
format long e;

% Declaramos las variables globales
global sigmax; % Coeficiente asociado al término de difusión
global sigmay; % Coeficiente asociado al término de difusión
global sigmaz; % Coeficiente asociado al término de difusión
global Qs; % Término fuente
global u; % Velocidad del viento
global xs; % Posición x para generar las medidas sintéticas
global ys; % Posición y para generar las medidas sintéticas
global zs; % Posición z para generar las medidas sintéticas
global xM1; % Posición x del primer sensor
global yM1; % Posición y del primer sensor
global zM1; % Posición z del primer sensor
global xM2; % Posisión x del segundo sensor
global yM2; % Posición y del segundo sensor
global zM2; % Posición z del segundo sensor
global xM3; % Posisión x del tercer sensor
global yM3; % Posición y del tercer sensor
global zM3; % Posición z del tercer sensor
global xM4; % Posisión x del cuarto sensor
global yM4; % Posición y del cuarto sensor
global zM4; % Posición z del cuarto sensor
global tM1; % Tiempo de medida 1
global tM2; % Tiempo de medida 2
global tM3; % Tiempo de medida 3
```

```
global tM4; % Tiempo de medida 4
global DHD; % Distancia de riesgo a sotavento
global icaso; % Caso que resolvemos

% Llamamos a la función de datos
datos;

% Generamos las variables sigma
sigma();

% Definimos otras variables para la representación
tMR=60.0*10; % Tiempo para la representación
HR=2.0;  % Altura para la representación
N=30; % Número de divisiones en la malla
LMAX=DHD; % Dimensión del dominio

% Generamos la lista de concentraciones para la representación
clist = [1e-5, 1e-4, 1e-3, 1e-2, 0.025, 0.05, 0.1];

% Generamos la malla
[xx,yy] = meshgrid([0:N]*LMAX/N,[0:N]*LMAX/N);

% Generamos el vector de tiempos
tiempos=linspace(0,tMR,100);

% Calculamos el término constante
ctemp=Qs/(sqrt(2*pi*sigmax^2)*sqrt(2*pi*sigmay^2)*sqrt(2*pi*sigmaz^2));

% Realizamos un bucle en los pasos de tiempo
for tt=tiempos

    % Calculamos el valor de la concentración en los nodos del mallado
    cc=ctemp*exp(-(xx-xs-u*tt).^2/(2*sigmax^2)).*...
             exp(-(yy-ys).^2/(2*sigmay^2)).*...
            (exp(-(HR-zs).^2/(2*sigmaz^2))+exp(-(HR+zs).^2/(2*sigmaz^2)));

    % Realizamos la representación gráfica
    [cs,ch] = contourf(xx,yy,cc);
    hold on
    if(icaso==1)
    p1=plot(xM1,yM1,'*','MarkerSize',10,'MarkerEdgeColor','b');
    p2=plot(xM2,yM2,'*','MarkerSize',10,'MarkerEdgeColor','g');
    end
    if(icaso==2)
    p1=plot(xM1,yM1,'*','MarkerSize',10,'MarkerEdgeColor','b');
    p2=plot(xM2,yM2,'*','MarkerSize',10,'MarkerEdgeColor','g');
    p3=plot(xM3,yM3,'*','MarkerSize',10,'MarkerEdgeColor','m');
    end
    if(icaso==3)
    p1=plot(xM1,yM1,'*','MarkerSize',10,'MarkerEdgeColor','b');
    p2=plot(xM2,yM2,'*','MarkerSize',10,'MarkerEdgeColor','g');
    p3=plot(xM3,yM3,'*','MarkerSize',10,'MarkerEdgeColor','m');
    p4=plot(xM4,yM4,'*','MarkerSize',10,'MarkerEdgeColor','k');
    end
    colormap jet
    clabel(cs)
    title(['Tiempo actual ',num2str(tt),' (s)']);
    xlabel('Posición eje x (m)')
```

```
    ylabel('Posición eje y (m)')
    c = colorbar;
    c.Label.String = 'Concentración de agente químico (kg/m^3)';
    if(icaso==1)
    legend([p1 p2],'Sensor nº1','Sensor nº2')
    end
    if(icaso==2);
    legend([p1 p2 p3],'Sensor nº1','Sensor nº2','Sensor nº3')
    end
    if(icaso==3);
    legend([p1 p2 p3 p4],'Sensor nº1','Sensor nº2','Sensor nº3','Sensor nº4')
    end
    pause
    hold off;
end
```

AI.12 mallado.m

```
%*************************************************************************
% Autor: Juan Manuel Ortega Muñiz
% Nombre de la función: mallado
% Descripción: pequeño script para estudiar el impacto de la posición de
% la fuente de emisión
%*************************************************************************

% Limpiamos el workspace
clear global
clear all
format long e;

% Declaramos las variables globales que emplearemos
global Dx; % Distancia en el eje x en m
global Dy; % Distancia en el eje y en m
global xs; % Posicion x para generar las medidas sintéticas
global ys; % Posicion y para generar las medidas sintéticas
global zs; % Posicion z para generar las medidas sintéticas

Dx=1300;
Dy=1300;

% Llamamos a la función de datos
datos;

% Generamos los valores de sigma
sigma;

% Opciones para el algoritmo de optimización
options=optimset('Jacobian','on','Display','iter','MaxIter',50,...
    'TolFun',1.e-21,'MaxFunEvals',1000);

% Punto inicial
x0=[325;325;2.5];

% Cotas inferiores para el punto de vertido
lb=[0;0;0];

% Cotas superiores para el punto de vertido
```

```
ub=[Dx;Dy;5];

% Bucle para variar el iterante inicial
% Dividimos bidimensionalmente el entorno de trabajo
n=100; % Número de divisiones en el mallado
x=linspace(0,Dx,n);
y=linspace(0,Dy,n);

% Definimos las variables de salida del algoritmo
xsol=zeros(n,n,3);
Posic_fuente=zeros(n,n,3);

resnorm=zeros(n,n);
%residual=zeros(n,n);

cont=0;
for i=1:n
    for j=1:n
        cont=cont+1;
        % Generamos las medidas
        xs=x(i);
        ys=y(j);
        medidas;

        % Guardamos las posiciones de la fuente
        Posic_fuente(i,j,1)=xs;
        Posic_fuente(i,j,2)=ys;
        Posic_fuente(i,j,3)=zs;

        disp('Iteración:')
        disp(cont);

        % Lanzamos el algoritmo
        [xsol(i,j,:)]=lsqnonlin(@EvalCoste,x0,lb,ub,options);
    end
end

% Calculamos el error que comete el algoritmo
Errores=abs(Posic_fuente-xsol);

for i=1:n
    for j=1:n
        vec_aux=[];
        vec_aux=[vec_aux Errores(i,j,1)];
        vec_aux=[vec_aux Errores(i,j,2)];
        % Almacenamos solo la norma de las dos primeras componentes del
        % error pues la altura z ya sabemos que no importa frente a x e y
        Errores_norma(i,j)=norm(vec_aux);
    end
end

figure
subplot(2,2,1)
surf(x,y,Errores_norma(:,:))
title('Norma de los errores en (X,Y)')
axis([0 Dx 0 Dy 0 600])

subplot(2,2,2)
```

```
surf(x,y,Errores(:,:,1))
title('Errores en la dirección X')
axis([0 Dx 0 Dy 0 600])

subplot(2,2,3)
surf(x,y,Errores(:,:,2))
title('Errores en la dirección Y')
axis([0 Dx 0 Dy 0 600])

subplot(2,2,4)
surf(x,y,Errores(:,:,3))
title('Errores en la dirección Z')
axis([0 Dx 0 Dy 0 600])

% Mostramos información del resultado
disp('######## FIN DE LA SIMULACION ########')
```