UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

“Simulación fotoacústica con k-Wave”

TRABAJO FINAL DE GRADO

Autor/a:

Luis Núñez Rodríguez

Tutor/a:

Francisco Camarena Femenía

GANDIA, 2016

Resumen

Este trabajo tiene como objetivo el estudio del efecto fotoacústico para imagen biomédica empleando la herramienta de simulación basada en Matlab *k-Wave*. En primer lugar se realizará una introducción al efecto fotoacústico y los conceptos teóricos más relevantes para su estudio. A continuación, se explicará el funcionamiento de la herramienta *k-Wave* y los parámetros necesarios para la simulación. Por último se expondrán las simulaciones realizadas con sus respectivos resultados.

Abstract

This project has the aim of studying the photoacoustic effect for biomedical imaging with Matlab based toolbox “k-Wave”. Firstly, photoacoustic effect and theoretical relevant concepts will be introduced. Then operation with k-Wave toolbox and parameters required for the simulation will be explained. Finally simulations and their respective results will be presented.

Palabras clave

Fotoacústica, ultrasonidos, k-Wave, Matlab, simulación, imagen biomédica

Keywords

Photoacoustic, ultrasound, k-Wave, Matlab, simulation, biomedical imaging

Índice

Índice de figuras

Simulación fotoacústica con k-Wave

1. Efecto fotoacústico

En este capítulo se realiza una introducción en el contexto histórico del efecto fotoacústico, sobre el cual, también se comentarán ciertos conceptos teóricos necesarios para su estudio.

1.1. Introducción histórica

El efecto fotoacústico fue descubierto hace más de 100 años (1880) por Alexander Graham Bell al detectar que una señal acústica puede producirse iluminando con radiación modulada periódicamente una muestra colocada en una celda cerrada.

Bell trabajaba junto a Charles Sumner Tainter en la invención del fotófono, que permitiría transmitir la voz a grandes distancias a través de la luz solar. Para esto, experimentaba reflejando un haz de luz solar sobre una celda de selenio incorporada a un circuito telefónico. El haz era reflejado a través de un espejo colocado en el diafragma de un objeto similar a un altavoz que vibraba al ser activado por la voz. La resistencia eléctrica del selenio era modulada entonces por la luz, reproduciéndose la voz transmitida hacia el recibidor telefónico.

Este invento terminó fracasando, ya que solo permitía transmitir la voz a unas decenas de metros y no funcionaba en días nublados pero resultaba interesante experimentalmente. Bell continuó investigando sobre las propiedades del selenio, demostrando que este material emite sonido cuando es iluminado por luz modulada. También descubrió que la intensidad del sonido emitido dependía de la longitud de onda de la luz incidente, atribuyendo el efecto a la absorción óptica del material.

A pesar de estos descubrimientos, no fue hasta casi un siglo después (1960) cuando el desarrollo del micrófono, la aparición de los láseres y la mejora de los sistemas de detección y procesamiento de datos, permitieron la aparición de las primeras aplicaciones de esta técnica. Más adelante a mediados de 1990 fue cuando comenzó a investigarse su aplicación para imagen biomédica, de la que se va a tratar en este trabajo.

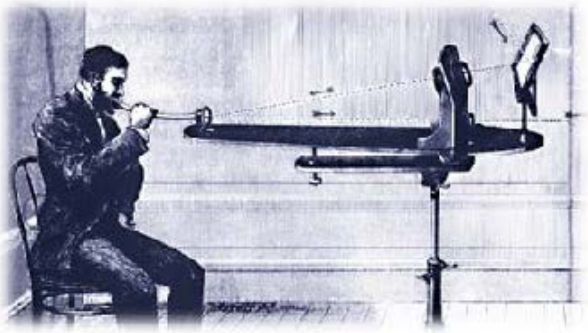

Figura 1: Fotófono de Graham Bell

1.2. Conceptos teóricos

A continuación se explicarán los conceptos teóricos más relevantes para comprender el efecto fotoacústico.

1.2.1. Generación de la señal fotoacústica

El proceso de generación de la señal fotoacústica viene originado por unas ondas lumínicas emitidas por un láser que inciden en la superficie de la piel. Dependiendo del tamaño de las ondas, la luz penetrará a una determinada profundidad. Al incidir, estas ondas son dispersadas y absorbidas por unas moléculas específicas de la piel llamadas cromóforos (como la melanina o la hemoglobina). La energía lumínica es convertida en calor mediante vibraciones y colisiones que provocan una pequeña subida de temperatura inferior a la necesaria para causar daños físicos o cambios fisiológicos en la piel. Esto produce un incremento de la presión inicial (p0) y la consecuente relajación, resultando en la emisión de ondas acústicas de baja amplitud. Dichas ondas se propagan a la superficie donde son detectadas por un transductor o un array de transductores ultrasónicos.

El proceso antes descrito, puede ser visto como un mecanismo en el que la distribución de la presión inicial (p0) es codificada en una onda acústica, que tras su detección por un transductor situado en la superficie, será convertida en una señal eléctrica. Por lo que se puede decir que la imagen fotoacústica es una representación de la presión inicial (p0).

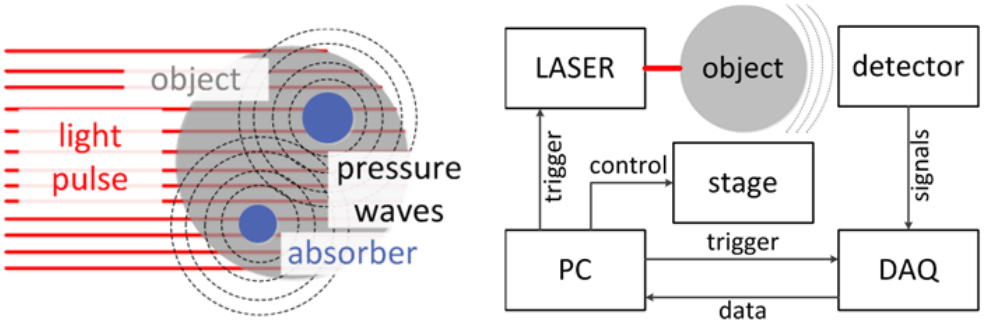

Figura 2: Esquema de generación de señal FA

1.2.2. Ecuación de presión inicial

La presión inicial está relacionada con el calor producido por la energía lumínica que proviene del láser. Suponiendo que el calor generado se propaga en un intervalo de tiempo mayor que el tiempo de propagación acústica (ms > μs), y que a su vez será mucho mayor que el intervalo de tiempo del pulso láser (ms >> ns), entonces se puede considerar que el calor no será conducido a los puntos cercanos y que la expansión de volumen es casi insignificante, por lo que p0 es proporcional en un punto r a la energía óptica absorbida [2]:

$$p0 = \Gamma H(r) = \frac{\beta c^2}{C_p} \cdot H(r)$$

La constante Γ es el coeficiente de Grüneisen que relaciona β (volumen de dilatación térmica), c (velocidad del sonido) y C_p (capacidad calorífica) proporcionando la conversión de energía calorífica a presión. Por otra parte H (r) es la densidad de energía óptica absorbida que es el producto del coeficiente de absorción (μ_a) y la fluencia óptica (ϕ) que indica la tasa de energía que está siendo aplicada en el tejido y es dependiente del propio coeficiente de absorción (μ_a), del coeficiente de dispersión (μ_s) y del factor de anisotropía (g):

$$p_0(r) = \Gamma \mu_a(r) \cdot \phi(r; \mu_a; \mu_s; g)$$

Si se analizan los términos de la ecuación resultante, se puede ver que la presión inicial dependerá de una serie de parámetros mecánicos, termodinámicos y ópticos. Sin embargo, en el estudio del efecto fotoacústico, las propiedades mecánicas y termodinámicas se consideran prácticamente invariantes en diferentes tipos de tejidos. Por lo que puede deducirse que el contraste de la imagen fotoacústica resultante de la representación de p0 estará dominado por las propiedades de absorción y dispersión óptica del tejido.

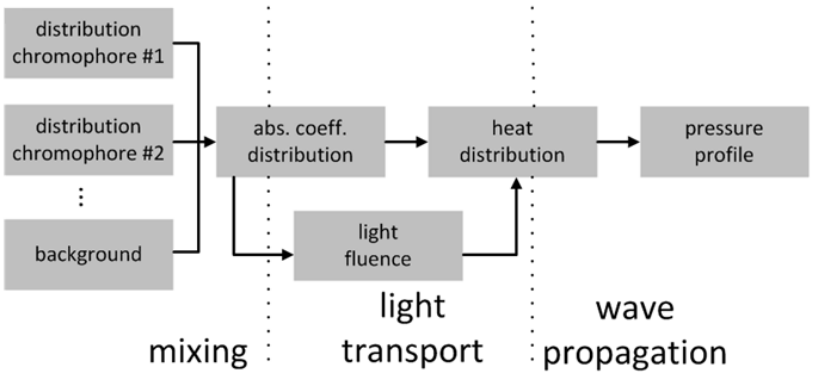

Figura 3: Factores de la ecuación p_0

2. K-Wave

En este capítulo se van a tratar los aspectos más relevantes de la herramienta k-Wave con la cual se realizan las simulaciones en Matlab del efecto fotoacústico, con una introducción a su funcionamiento y los parámetros a definir en las simulaciones.

2.1. Introducción

K-Wave es una herramienta para Matlab de código abierto diseñada para la simulación en dominio temporal de la propagación de ondas acústicas. Se trata de un modelo numérico avanzado que permite simular la propagación lineal y no lineal de las ondas, los parámetros de materiales heterogéneos y la absorción acústica. Esta herramienta fue desarrollada en el Photoacoustic Imaging Group del University College London [5].

2.1.1. Ecuaciones de onda

La evolución temporal de los campos de ondas fotoacústicas puede ser modelada usando las ecuaciones de acústica lineal. Cuando una onda acústica pasa por un medio compresible, hay fluctuaciones dinámicas en la presión, densidad, temperatura, velocidad de las partículas, etc. Estos cambios pueden ser descritos por un conjunto de ecuaciones diferenciales parciales de primer orden basadas en la conservación de la masa, del momento y la energía en el medio. En el caso de una onda acústica en un medio fluido homogéneo y sin pérdidas, las ecuaciones serían:

$$\frac{\delta u}{\delta t} = -\frac{1}{\rho_0}\nabla p \quad \text{(Conservación del momento)}$$

$$\frac{\delta \rho}{\delta t} = -\rho_0 \nabla \cdot \mathrm{u} \text{ (Conservación de la masa)}$$

$$p = c_0^2 \rho \text{ (Relación presión - densidad)}$$

Donde u es la velocidad de partícula acústica, p es la densidad acústica (presión), ρ_0 es la densidad de ambiente y c_0 la velocidad del sonido. Estas ecuaciones asumen que el medio es inmóvil (los parámetros ambientales no cambian con el tiempo) e isotrópico (los parámetros materiales no dependen de la dirección a la que se mueve la onda). Estas ecuaciones se pueden combinar para formar una única ecuación de onda de segundo orden con una única variable acústica (normalmente la presión):

$$\nabla^2 p - \frac{1}{c_0^2} \cdot \frac{\delta^2 p}{\delta t^2} = 0$$

En el caso de las simulaciones con k-Wave, se suele resolver el sistema de ecuaciones de primer orden en vez de la ecuación de segundo orden equivalente. Esto es debido a que es más simple introducir las fuentes de fuerza y masa en las ecuaciones, los valores de presión y velocidad de partículas serán calculados de forma más precisa al emplearse una malla y también permite el uso de una capa especial llamada PML, que absorbe las ondas acústicas cuando alcanzan los límites del dominio computacional.

2.1.2. Método de resolución de ecuaciones

Hay gran variedad de métodos numéricos para resolver las ecuaciones diferenciales parciales. El más idóneo para un problema concreto depende de muchos factores a tener en cuenta. En el caso de la fotoacústica se busca una solución en dominio temporal a la ecuación de onda para ondas acústicas de banda ancha. El método clásico de diferencias finitas (FD) emplea una malla con al menos 10 puntos por tamaño de onda para un buen nivel de precisión. Esto provoca que las mallas computaciones sean demasiado grandes para resolverlas en ordenadores normales. Para reducir la memoria y el intervalo de tiempo, k-Wave resuelve el sistema de ecuaciones acústicas, que antes se ha comentado, usando el método pseudo-espectral de espacio-k.

El método de espacio-k combina el cálculo espectral de las derivadas parciales mediante el método de colocación de Fourier. En el método de Diferencias Finitas (FD), los gradientes espaciales son calculados localmente basados en los valores de la función de sus puntos vecinos de la malla (interpolación lineal). Por lo que cuantos más puntos se usen más preciso será el método.

En el método de espacio-k se ajusta una serie de Fourier a todos los datos, considerándolo un método global en vez de local. Al usar series de Fourier las amplitudes de sus componentes se pueden calcular con la transformada FFT y al ser funciones sinusoidales, solo se necesitarán dos puntos de malla por tamaño de onda.

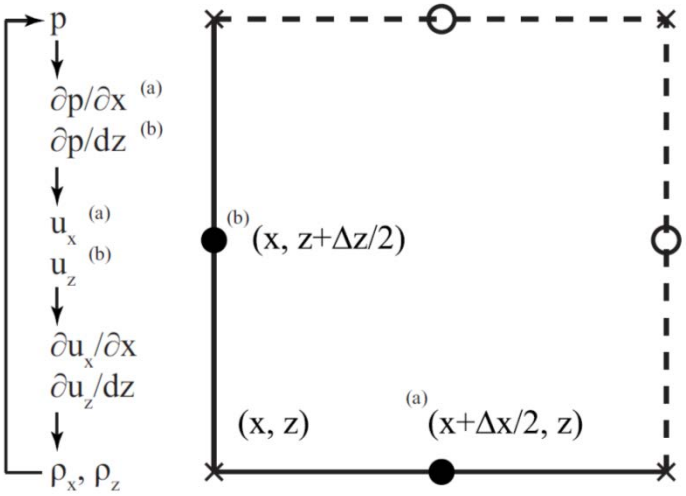

Figura 4: Esquema del método espacio-k

2.1.3. Perfectly Matched Layer

Las simulaciones de campos de ondas propagatorios utilizan mallas computacionales de tamaño finito, por lo que requieren un esquema numérico eficiente para calcular las derivadas cerca de los bordes de la malla. En el caso del método espacio-k, el cálculo de las derivadas espaciales mediante FFT provoca que las ondas salgan por una parte del dominio para reaparecer en el lado opuesto. Para modelar la propagación de estas ondas en el espacio libre se puede incrementar el tamaño de la malla hasta que las ondas nunca alcancen los límites, provocando que sea más complicado de calcular.

Este fenómeno puede ser evitado implementando una capa de absorción en los límites conocida como Perfectly Matched Layer (PML) [5]. Se trata de una fina capa absorbente que envuelve el dominio computacional y se rige por un conjunto de ecuaciones no físico que provoca atenuación anisotrópica. Esta capa debe cumplir dos requisitos, que proporcione suficiente absorción para que las ondas salientes sean significantemente atenuadas y que no refleje ninguna onda al medio.

El uso de esta capa requiere que la presión sea dividida artificialmente en componentes cartesianas ($p = p_x + p_y + p_z$). Por lo tanto la absorción será entonces definida tal que solo las componentes de la onda que viajen a través de la PML y dentro de los límites sean absorbidas. Por lo tanto incluyendo la PML, las ecuaciones acústicas de primer orden serían las siguientes:

$$\frac{\delta u}{\delta t} = -\frac{1}{\rho_0}\nabla p - \alpha \cdot u$$

$$\frac{\delta \rho_x}{\delta t} = -\rho_0 \frac{\delta u_x}{\delta x} - \alpha_x \rho_x$$

$$p = c_0^2 \cdot \Sigma \rho_{x,y,z}$$

Donde $\alpha = \{\alpha_x, \alpha_y, \alpha_z\}$ es la absorción anisotrópica en Nepers por metro, que únicamente no es igual a cero dentro de la PML. El rendimiento de la PML depende del tamaño y la atenuación de la capa, así como del intervalo de tiempo usado en la simulación. Cuando está bien implementada, es posible simular el dominio infinito de propagación usando una pequeña malla computacional.

2.2. Definición de parámetros

Se utilizan scripts de Matlab para definir las funciones a realizar por las simulaciones de la herramienta k-Wave. Estas funciones constan de cuatro estructuras de entrada a definir como son la malla (*kgrid*), el medio (*medium*), la fuente (*source*) y el sensor (*sensor*). Los parámetros y propiedades de cada estructura se definen en campos con la forma *structure.field*. Cuando se ejecuta la función de simulación (kspaceFirstOrder2D en el caso de dos dimensiones), la propagación de las ondas en el medio es calculada paso por paso, guardando los campos acústicos en los elementos del sensor después de cada instante, mostrando los valores cuando el bucle de tiempo se termina.

2.2.1. Malla

La primera estructura llamada *kgrid* define las propiedades de la malla computacional, determinando cómo estará dividido el medio en forma malla de puntos (nodos) a modo de cuadrícula. Cada punto de la malla representa una posición donde se van a resolver las ecuaciones.

Para crear esta malla hay que utilizar la función *makeGrid* con una variable de entrada para indicar el número de puntos (Nx, Ny, Nz) y otra para el espaciado entre puntos (dx, dy, dz) en cada dirección cartesiana. Estas variables son usadas para crear las matrices de número de onda[1] y coordenadas cartesianas de la malla. Los números de onda se emplean para calcular los gradientes espaciales de los parámetros del campo acústico usando el método de colocación espacial de Fourier, anteriormente comentado.

Los cálculos del gradiente espacial usados en k-Wave, hacen uso de la transformada rápida de Fourier (FFT). Dependiendo de la complejidad de la simulación, puede incluso llegar a calcularse 14 FFT por cada instante de tiempo. El tiempo para calcular cada FFT puede ser minimizado eligiendo el número total de puntos de malla en cada dirección como un múltiplo de 2 o con factores primos pequeños.

En cuanto a la capa PML, normalmente suele ocupar una zona de 20 puntos de la malla alrededor del eje del dominio. Es importante situar la capa fuera de la malla para que no se absorban las ondas dentro del dominio computacional de nuestra simulación.

Por último, también resulta necesario mencionar el parámetro de la matriz de tiempo *t_array*. Este parámetro, indicará los valores de tiempo en los cuales se realizará la simulación y por defecto se suele crear de forma automática con la función *makeTime*. El tiempo que tarda la onda acústica en desplazarse a través de la diagonal mayor de la malla a la mínima velocidad de sonido determinará el tiempo total del array. Mientras que el intervalo de tiempo se calcula basándose en la máxima velocidad de sonido en el medio, el tamaño de la malla y el número de Courant-Friedrichs-Lewy (CFL) con valor de 0.3:

$$\Delta t = CFL * \frac{\Delta x}{c_{max}}$$

[1] Número de onda: Magnitud de frecuencia que indica el número de veces que vibra una onda en una unidad de distancia.

2.2.2. Medio

La segunda estructura a definir *medium* define las propiedades materiales del medio en cada punto de la malla. Hay 5 propiedades materiales que pueden ser definidas: la velocidad de sonido (*sound_speed*), la densidad de masa (*density*), el parámetro de no linealidad (*BonA*), el coeficiente de absorción (*alpha_coeff*) y el exponente de absorción (*alpha_power*). Dependiendo de si los parámetros son heterogéneos, pueden ser definidos como escalares o matrices del tamaño de la malla, excepto el exponente que será escalar.

El parámetro de velocidad de sonido deber ser definido en todas las simulaciones, la densidad solo puede ser omitida en medios homogéneos y sin pérdidas, mientras que los otros tres parámetros son opcionales. Si no se define la no linealidad, se asume que la simulación será lineal, al igual que si los parámetros de absorción no son definidos se asumirá que la simulación es sin pérdidas.

Los parámetros de absorción corresponden a la ley de absorción exponencial definida tal que $\alpha = \alpha_0 f^y$ siendo α_0 el coeficiente de absorción (*alpha_coeff*) en unidades de $dB\ MHz^{-y} cm^{-1}$ y el exponente de absorción (*alpha_power*) representado por y con valores entre $0 \leq y \leq 3$ y no igual a 1. La absorción se modela con estos dos términos que siguen una ley potencial de frecuencia, tal y como se puede observar en la siguiente gráfica:

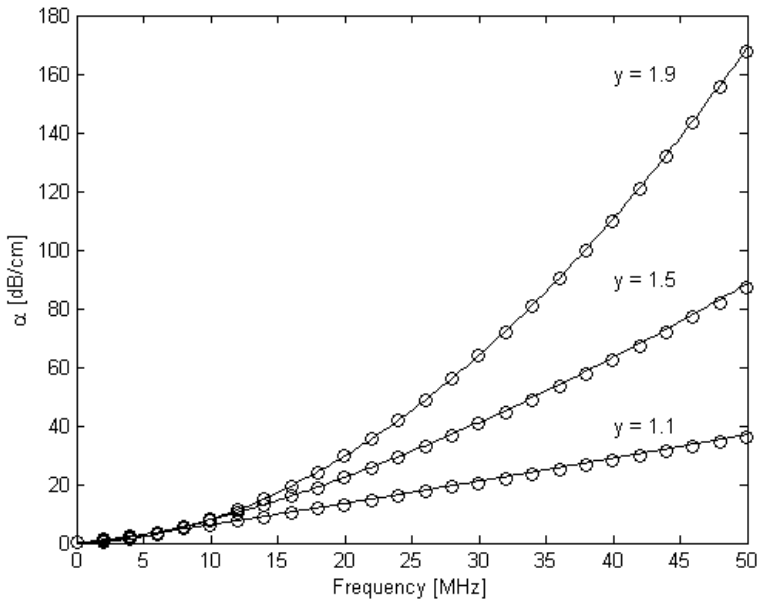

Figura 5: Ley de absorción exponencial

2.2.3. Fuente

La tercera estructura define las propiedades y ubicación de cualquier fuente acústica en el medio. Hay tres tipos de fuentes que se pueden emplear, una distribución inicial de presión, una fuente de presión variante en el tiempo y una fuente de velocidad de partículas variante en el tiempo. La fuente más apropiada para las simulaciones que se van a realizar es la distribución de presión inicial, ya que es posible modelarla con la forma que más se ajuste a lo que se pretende simular. Esta fuente se define mediante la asignación de una matriz a la variable *source.p0*. Las únicas condiciones que debe cumplir son que la matriz tenga el mismo tamaño que la malla computacional y que los valores sean reales. En la herramienta se incluyen varias funciones para crear formas geométricas como círculos, discos, líneas o esferas que representan los puntos donde se generará la presión inicial.

Por defecto la distribución de presión inicial *p0* es suavizada mediante la función s*mooth* antes de comenzar la simulación. Se suele emplear un filtro de ventana en dominio frecuencial (por defecto Blackman) para reducir las oscilaciones generadas debido al uso del método de colocación espectral de Fourier. Estas se generan porque en cada punto de la malla se obtiene un coeficiente de Fourier y con una función continua los puntos son interpolados con componentes de Fourier. Cuando hay grandes saltos entre puntos adyacentes, aparecen oscilaciones al unirlos tal y como se puede ver en la siguiente gráfica:

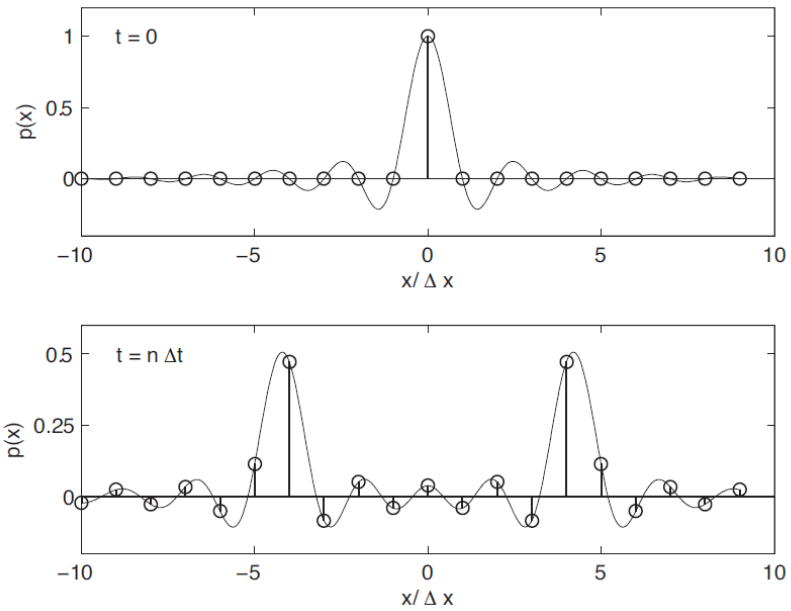

Figura 6: Oscilaciones en la simulación

Para reducir estas oscilaciones visibles, se disminuirá la amplitud de las componentes frecuenciales más altas, haciendo que el tamaño de saltos entre puntos de malla consecutivos sea más pequeño. Se puede ver un ejemplo en la siguiente figura:

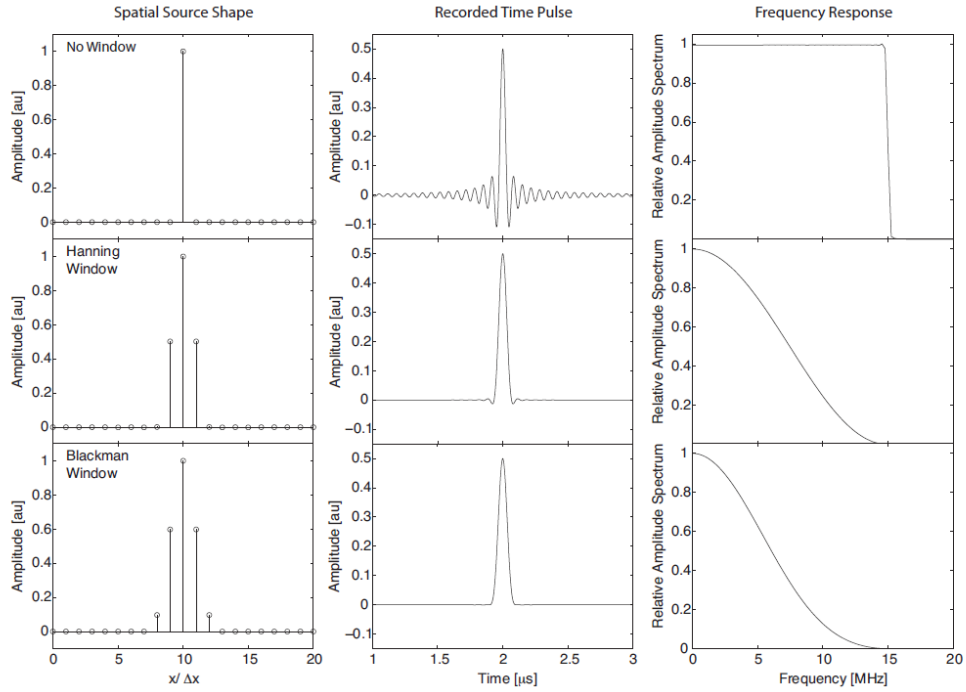

Figura 7: Comparativa del uso de smooth

2.2.4. Sensor

La última estructura de entrada define las propiedades y la ubicación de los puntos del sensor que guarda los campos acústicos a cada instante de la simulación. La posición de los puntos del sensor en el dominio computacional es definida con el parámetro *mask*. Se puede emplear una matriz binaria o unas coordenadas cartesianas que especificarán los puntos de la malla donde se guardan los datos.

La máscara binaria se define asignando una matriz con el mismo tamaño que la malla computacional, donde los 1 representan los puntos de la malla dentro del dominio que forman parte del sensor. En cambio el sensor cartesiano se define con una matriz N x M de coordenadas, donde N es el número de dimensiones y M el número de puntos del sensor.

Cuando se utiliza una máscara de sensor cartesiana, los valores de los campos acústicos en cada punto se obtienen en cada instante de tiempo mediante interpolación lineal. Esto incrementará el tiempo de cálculo con respecto a la máscara lineal.

En cada instante de tiempo los valores de la presión acústica en los puntos del sensor son guardados en la variable *sensor_data* que será generada al terminar la simulación. También es posible guardar otras variables acústicas en el sensor como puede ser velocidad de partículas, intensidad acústica o presión RMS entre otras.

3. Simulación

En este tercer apartado, se van a explicar los tipos de reconstrucción de la imagen fotoacústica que permite utilizar k-Wave y la simulación en 2D realizada para modelar los posibles resultados de un sistema fotoacústico real a construir en el laboratorio.

3.1. Tipos de reconstrucción

3.1.1. FFT

El primer método de reconstrucción de imagen fotoacústica empleado en k-Wave realiza una estimación de la distribución de presión inicial guardada en el sensor mediante un algoritmo basado en FFT. La matriz de presión debe ir en formato *p_ty* (instante de tiempo, posición *y* de sensor). Emplea la función *kspaceLineRecon* para hacer la reconstrucción.

Este algoritmo realiza primero una transformada de Fourier de los datos en el espacio nº de onda – frecuencia a lo largo de las dimensiones *t* e *y*. En segundo lugar, se realiza el mapeo, basado en la relación de dispersión para una onda plana en un medio acústicamente homogéneo, de dominio nº de onda-frecuencia a dominio nº de onda-nº de onda. Por último se realiza la transformada de Fourier de vuelta al dominio espacial. El resultado es una estimación de la distribución inicial de presión acústica.

Los pasos 1 y 3 pueden ser realizados eficientemente usando una FFT pero el mapeo del paso 2 requiere una interpolación de los datos entre un dominio temporal y espacial. Se suele utilizar la interpolación *nearest* (vecino más cercano) por defecto para optimizar la velocidad de la simulación pero se puede emplear interpolación lineal que incrementa el tiempo para la reconstrucción pero reduce errores y ruido.

La resolución de la reconstrucción en la dirección y está definida por la ubicación y el espaciado de los elementos del sensor, mientras que en la dirección x está definida por la tasa de muestreo a la cual es registrada la presión.

La fotoacústica requiere que la presión acústica sea inicialmente positiva en todo el dominio pero la estimación de la distribución inicial de presión realizada con este método puede tener zonas negativas por los errores surgidos entre el modelo y la situación real. Para evitar esto se puede de manera opcional utilizar una condición de positividad *PosCond* que pondrá las zonas negativas a cero.

3.1.2. Time Reversal

El segundo método de reconstrucción de imagen fotoacústica consiste en utilizar la presión guardada por el sensor en la simulación, para asignarla por orden temporal inverso como una condición de frontera de Dirichlet sobre la superficie del sensor [6]. Para realizar esto, se deben asignar los datos de presión obtenidos en el sensor a la variable *sensor.time_reversal_boundary_data* y poner la fuente de presión acústica a cero. De esta forma, se realizará una reconstrucción más lenta pero más sencilla de aplicar, al solo tener que colocar los datos obtenidos directamente y utilizar la misma función que en la simulación (*kspaceFirstOrder2D* en el caso de dos dimensiones).

La reconstrucción Time Reversal lleva más tiempo de calcular pero se puede emplear en medios acústicos heterogéneos, con diferentes atenuaciones acústicas y máscaras de sensor con formas arbitrarias.

3.1.3. Ejemplos de reconstrucción de imagen

Para poder visualizar los métodos de reconstrucción, se va a mostrar un ejemplo simple con un sensor lineal en la parte de arriba de la malla y una distribución de presión inicial con forma círculo en el centro:

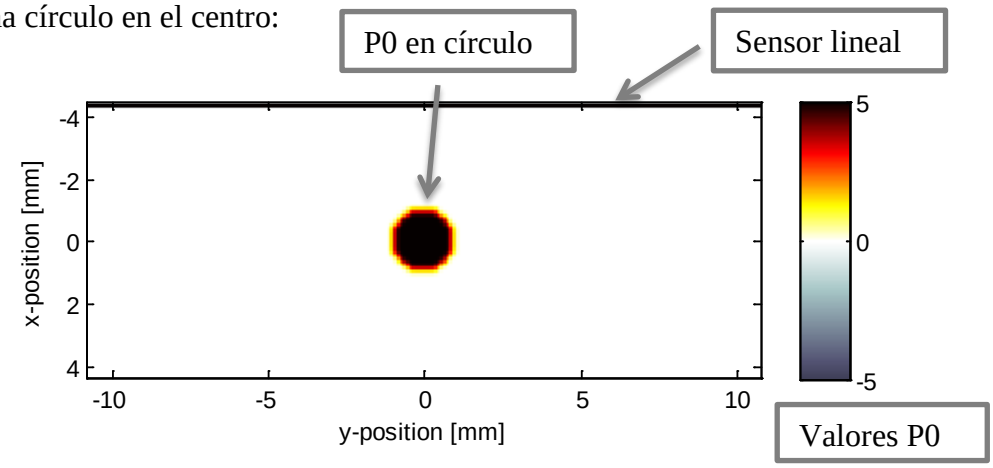

Figura 8: Ejemplo de condiciones iniciales de una simulación

Tras realizar la simulación, se mostrará un ejemplo de reconstrucción con el método de FFT y otro con el método de Time Reverse realizado con dicho ejemplo:

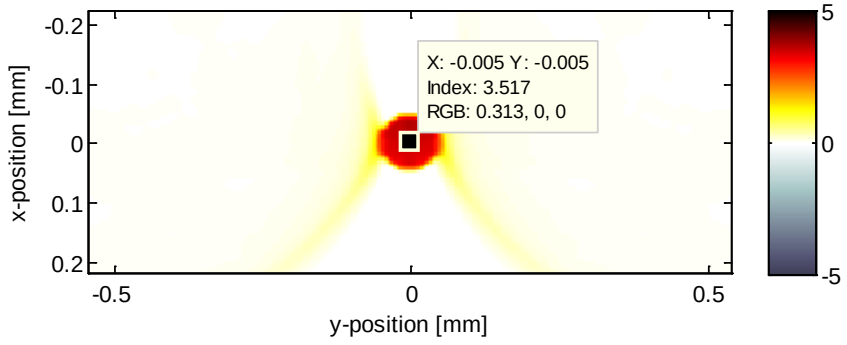

Figura 9: Reconstrucción simple FFT

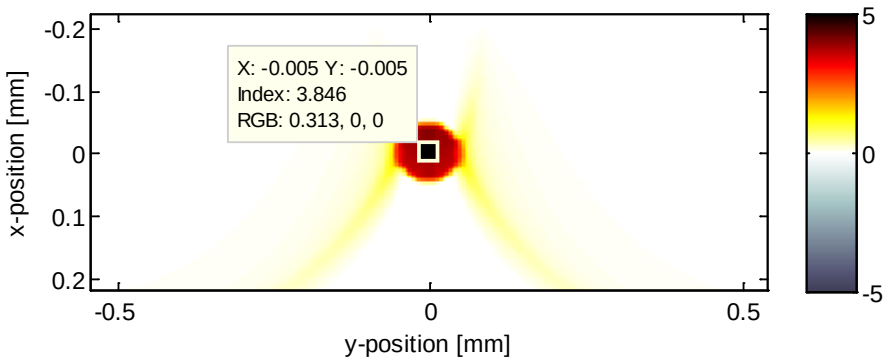

Figura 10: Reconstrucción simple con Time Reverse

En las imágenes se muestra una leyenda de colores para poder observar fácilmente los valores de presión iniciales y los obtenidos al reconstruir. Como se puede observar con el cursor situado en el punto (0,0) los valores en TR son ligeramente superiores que en FFT. En la siguiente gráfica comparativa se percibirá claramente esa diferencia en las reconstrucciones:

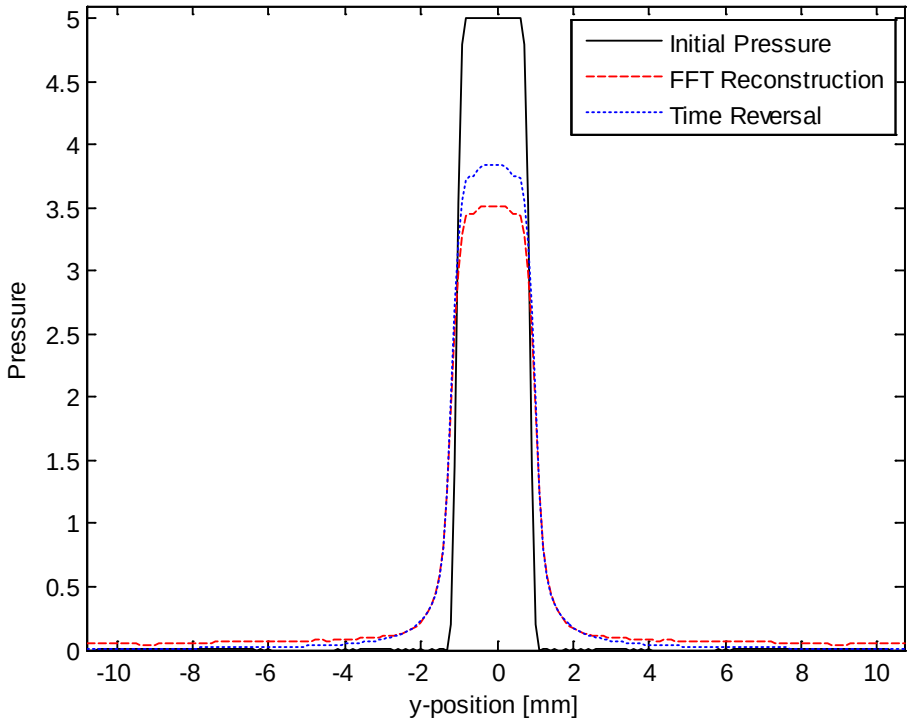

Figura 11: Comparativa FFT vs TR

Estos datos muestran que el método de reconstrucción Time Reverse obtiene resultados mejores en el mismo ejemplo, al aproximarse los valores de presión al supuesto inicial. Este ejemplo fue realizado en condiciones ideales y sin absorción en el medio. Además se ha cometido el llamado *inverse crime* que se explicará posteriormente.

3.2. Reconstrucción con sensor real

En este apartado se pretende estudiar las prestaciones del actual sistema de adquisición del que se dispone en el laboratorio mediante una simulación. En este caso se pretende usar un único transductor en vez de un *phased array*, para realizar la reconstrucción fotoacústica, por lo que se intentará comprobar si ésta será de buena calidad empleando el sensor en una única posición y también en varias.

3.2.1. Características del sensor y el phantom

Para la simulación a realizar se cuenta en el laboratorio con un sensor de marca Panametrics modelo V320-SU. Se trata de un transductor de inmersión de un único elemento longitudinal que tiene un frecuencia central de 7.5 MHz y un diámetro nominal del elemento de 13 mm. Para obtener mayor información sobre este transductor se ha empleado el analizador de impedancias disponible en el laboratorio y los resultados obtenidos se detallan en la siguiente gráfica:

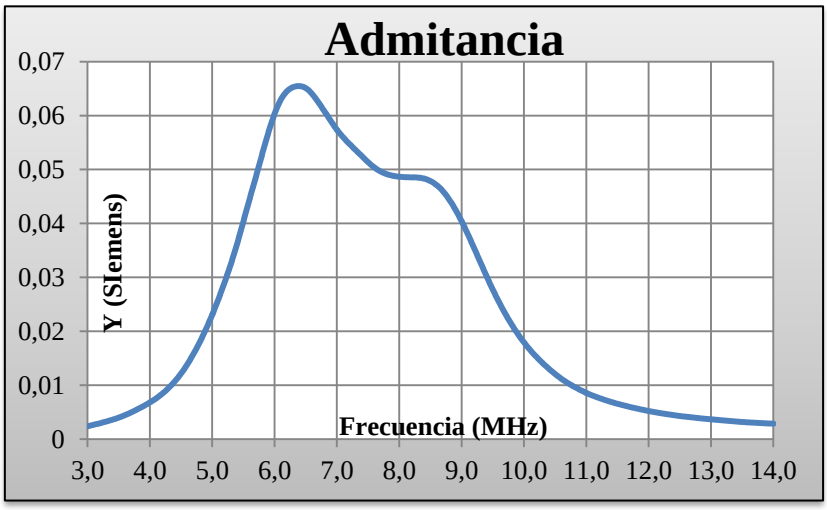

Figura 12: Gráfica admitancia con transductor V320-SU en aire

Tal y como se puede ver en la gráfica, las frecuencias de corte de este transductor se sitúa entre los 5 MHz y los 10 MHz, dando lugar a unos 5 MHz de ancho de banda. Por lo que el valor de ancho de banda en porcentaje será el resultado de la siguiente operación:

$$\frac{5MHz * 100}{7.5\ MHz} = 66.67\%$$

Para modelar la respuesta en frecuencia del sensor, se aplicará en la simulación un filtro de dominio Gaussiano a los datos obtenidos en la simulación (*sensor_data*) empleando la frecuencia central y el ancho de banda en porcentaje del transductor. Esto se realiza con la función *gaussianFilter* que le aplicará dicho filtro a cada fila de la matriz de los datos obtenidos. Esta función se realiza de forma automática al colocar los valores indicados de frecuencia central y ancho de banda en la variable *frequency_response* del sensor.

A modo de ejemplo, se pude observar a continuación el resultado de la simulación del ejemplo anterior con presión circular y sensor lineal. En este caso se ha introducido un sensor con frecuencia central de 7.5 MHz y ancho de banda de 5 MHz dando lugar a un porcentaje de 66.67 %, tal y como se ha comentado anteriormente. La gráfica muestra el nivel de presión en la posición del círculo de presión inicial, con la línea negra para la reconstrucción con un sensor ideal y la línea roja para la reconstrucción filtrada con los datos del sensor indicados:

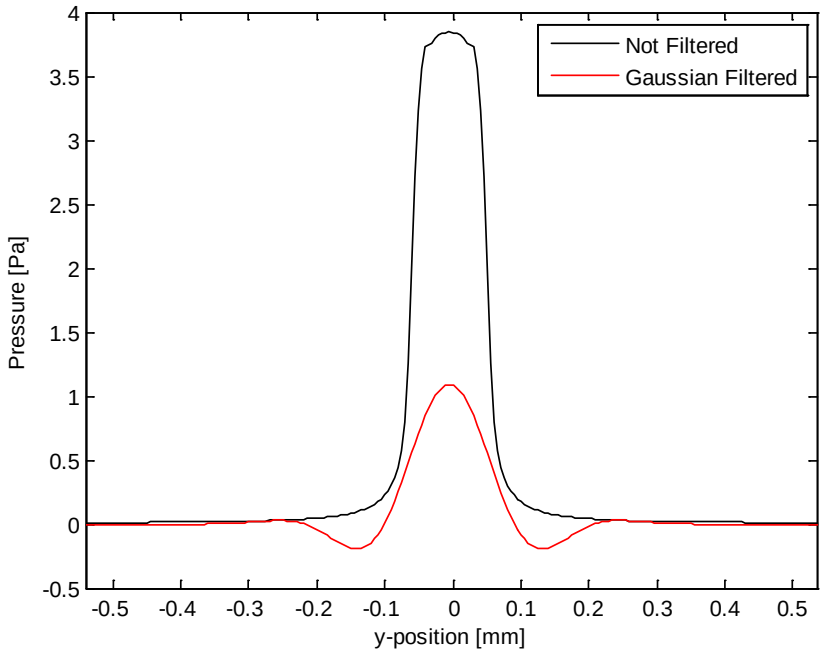

Figura 13: Ejemplo de respuesta en frecuencia del sensor tras reconstrucción

La simulación tratará de estudiar un futuro phantom con un tubo lleno de tinta que pretende representar una vena con sangre como si se realizara sobre la piel. Este phantom tendrá unas dimensiones de 20x20x30 mm y el tubo será de 20 mm de largo y 1.3 mm de diámetro. Se tratará de un caso similar al de la siguiente imagen obtenida del documento "Photoacoustic Imaging Using a Two-Dimensional CMUT Array" de los autores I. O. Wygant, X. Zhuang, P. S. Kuo, D. T. Yeh, O. Oralkan, B. T. Khuri-Yakub:

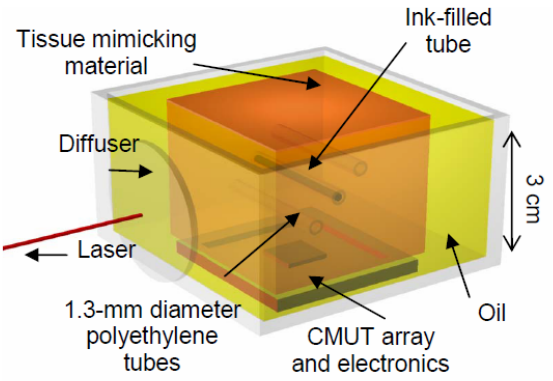

Figura 14: Dispositivo experimental en el que se basa la simulación

3.2.2. Sensor en una posición

El primer caso que se abordará en la simulación será observar los resultados que se podrían obtener construyendo el *phantom* y colocando el sensor en una única posición encima de este. Se ha elegido utilizar el método de reconstrucción Time Reversal por sus mejores resultados.

Para construir este *phantom* se comienza creando una malla del siguiente tamaño, 52 mm en la dirección X y 30 mm en la dirección Y. Se elige un tamaño de 20 puntos para la PML y 256 para la malla por ser factor primo de 2.

```
% Creación de la malla computacional
PML_size = 20;          % Tamaño en puntos de la PML
Nx = 256 - 2*PML_size;  % Nº de puntos en x (Fila)
Ny = 256 - 2*PML_size;  % Nº de puntos en y (Columna)
x = 52e-3;              % Tamaño malla en x [m]
y = 30e-3;              % Tamaño malla en y [m]
dx = x/Nx;              % Espaciado de puntos en x [m]
dy = y/Ny;              % Espaciado de puntos en y [m]
kgrid = makeGrid(Nx, dx, Ny, dy);
```

Para la distribución de presión inicial se ha elegido realizar unas líneas contiguas que después serán sumadas para dar lugar a una línea más gruesa que represente el tubo. Dado que el diámetro del tubo es de 1.3 mm se realiza una proporción teniendo en cuenta que en el eje Y 30 mm es igual a 216 puntos (256 menos 2*20 de la PML), por lo que 1.3 mm serán unos 9 puntos. Esto da lugar a 9 líneas que serán colocadas a lo largo del eje X centradas en el eje Y. Tras esto se realiza el suavizado y se asigna la distribución de presión inicial creada a la fuente.

```
%Presión inicial
magnitude = 1; % [Pa]
angle = 2*pi/2;     % [Radianes]
length = 83;    % [Puntos]

%Líneas de presión inicial
startpoint1 = [104 66];    % Punto de inicio [Puntos]
line1 = magnitude*makeLine(Nx, Ny, startpoint1, angle, length);
startpoint2 = [105 66];
line2 = magnitude*makeLine(Nx, Ny, startpoint2, angle, length);
                           ...
%Suma de líneas para hacer el tubo
line=line1+line2+line3+line4+line5+line6+line7+line8+line9;

% Suavizado de la presión inicial
p0 = smooth(kgrid, line, true);

% Asignar la presión a la fuente
source.p0 = p0;
```

El siguiente paso es definir las propiedades del medio. En este caso se va a suponer que se trata de un medio homogéneo y sin pérdidas, por lo que el único parámetro necesario por definir será la velocidad del sonido en el medio. En este caso utilizaremos 1500 m/s, ya que es la velocidad del sonido en el agua que será el medio natural donde se deberá realizar el futuro experimento y además se trata del medio con características más similares al tejido humano en cuanto a la velocidad de sonido.

```
% Propiedades del medio
medium.sound_speed = 1500;  % Agua [m/s]
```

En cuanto al sensor, como se ha comentado anteriormente, se trata de un sensor de 13 mm situado justo encima del *phantom*, por lo que será necesario de nuevo realizar una proporción tal que en el eje X 52 mm es igual a 216 puntos, por lo que 13 mm serán unos 54 puntos. Se elige el tamaño de 52 mm en el eje X porque se trata de cuatro veces el tamaño del sensor, que más adelante será empleado en otra simulación. Se intenta colocar el sensor centrado comenzando desde el punto 81. También se definirá la respuesta en frecuencia del sensor con los valores indicados del transductor V320-SU, el array de tiempo con la malla y la velocidad en el medio y los parámetros opcionales a elegir. En este caso se define que no se realice el suavizado automático que ya fue realizado, se sitúa la PML fuera de la malla y se le indica su tamaño y por último, se pide que no se muestre la PML en los gráficos de la simulación.

```
% Sensor lineal binario
sensor.mask = zeros(Nx, Ny);
sensor.mask(1,81:134) = 1;

% Array de tiempo
[kgrid.t_array, dt] = makeTime(kgrid, medium.sound_speed);

% Parámetros opcionales
input_args = {'Smooth', false, 'PMLInside', false,  'PMLSize', ...
PML_size, 'PlotPML', false};

% Respuesta en frecuencia del sensor
center_freq = 7.5e6;      % [Hz]
bandwidth = 66.67;        % [%]
sensor.frequency_response = [center_freq, bandwidth];
```

Una vez definidos todos los parámetros se pasa a realizar la simulación. En este caso al tratarse de un sensor con múltiples puntos (140) debido a su tamaño, k-Wave interpretará cada punto como un sensor diferente. Por lo que resulta necesario tratar los datos obtenidos para evitar esto. La solución tomada consiste en obtener la media de todos los datos obtenidos en cada punto buscando finalmente tener una única señal. Esta señal será repetida más adelante para obtener una matriz con mismo número de filas que de puntos del sensor. De esta manera, se realiza la reconstrucción mediante Time Reversal introduciendo como datos obtenidos la misma señal para cada uno de los puntos y así se simulará que ese conjunto de puntos es un único sensor.

```
% Realización de la simulación
sensor_data = kspaceFirstOrder2D(kgrid, medium, source, ...
sensor, input_args{:});

% Realizar la media de las señales obtenidas en el sensor
sensor_pres = sum(sensor_data,1)/sum(sensor.mask(:));
```

Uno de los efectos a tener en cuenta en este tipo de simulación con reconstrucción mediante el método *Time Reverse* es el *inverse crime*. Este efecto se produce al realizar los cálculos y después los cálculos inversos con los mismos parámetros y discretización escondiendo posibles errores en la simulación. Para evitar esto, se crea una malla diferente para la reconstrucción y también se le añade ruido a los datos obtenidos para alterar ligeramente la señal a reconstruir evitando emplear los mismos parámetros [6]. Se emplea la función *addNoise* que introducirá ruido Gaussiano aleatorio según el ratio de señal a ruido introducido y el nivel máximo de pico (*peak*) de la señal a tratar.

```
% Creación de otra malla
Nx = 256;           % Nº de puntos en x (Fila)
Ny = 256;           % Nº de puntos en y (Columna)
dx = x/Nx;          % Espaciado de puntos en x [m]
dy = y/Ny;          % Espaciado de puntos en y [m]
kgrid_recon = makeGrid(Nx, dx, Ny, dy);

% Añadir ruido a los datos obtenidos
signal_to_noise_ratio = 40; % [dB]
sensor_pres = addNoise(sensor_pres, signal_to_noise_ratio, 'peak');

% Repetir la media para cada punto del sensor
sensor_rep = repmat(sensor_pres,54,1);
```

Por último, se realiza la reconstrucción con la señal única repetida, asignando el time array de nuevo, definiendo de nuevo el sensor lineal, poniendo la presión inicial a cero para no repetir de nuevo la simulación y definiendo el resto de parámetros ya indicados.

```
% Sensor lineal binario para la reconstrucción
sensor_recon.mask = zeros(Nx, Ny);
sensor_recon.mask(1,81:134) = 1;

% Asignar array de tiempo original
kgrid_recon.t_array = kgrid.t_array;

% Presión inicial a cero
source.p0 = 0;

% Asignar datos para reconstrucción
sensor_recon.time_reversal_boundary_data = sensor_rep;

% Realización de la reconstrucción Time Reversal
p0_recon = kspaceFirstOrder2D(kgrid_recon, medium, source, ...
sensor_recon, input_args{:});
```

El primero de los gráficos a mostrar tras realizar la simulación, es el gráfico donde se puede observar la distribución de presión inicial en forma de tubo situada en el centro del eje Y a lo largo del eje X y la máscara del sensor en la parte de arriba del eje Y .

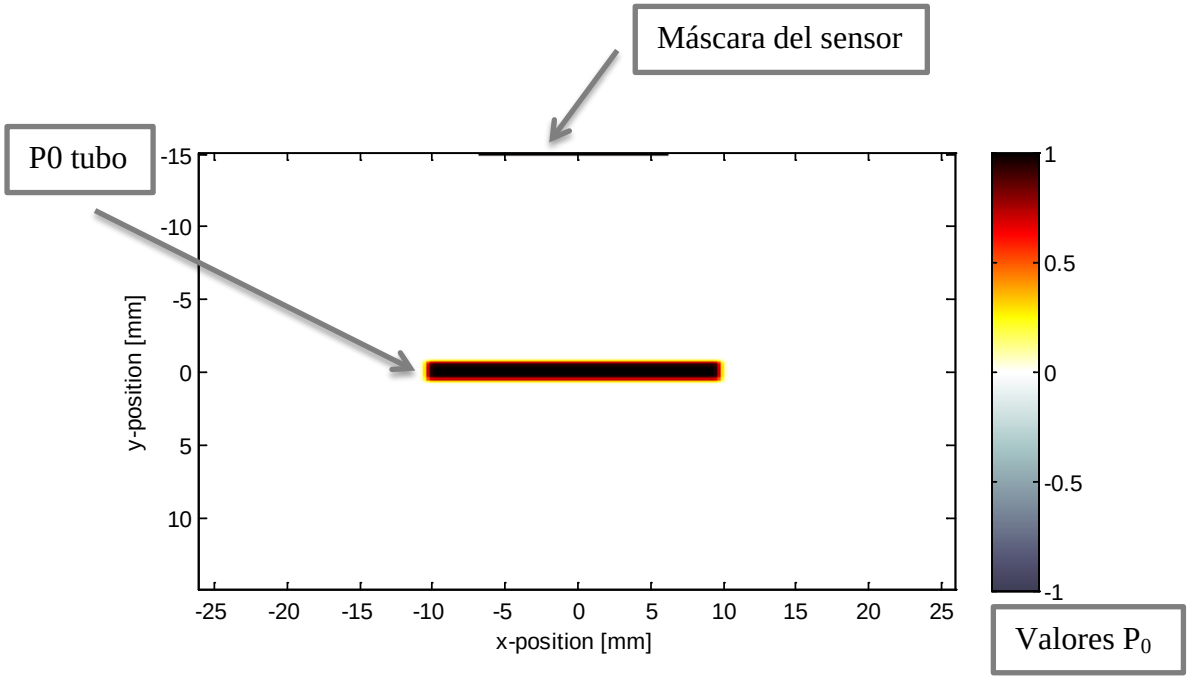

Figura 15: Situación inicial de reconstrucción con una posición

En el siguiente gráfico se muestra la onda recibida por el sensor en el intervalo de tiempo de la simulación. Se puede observar la forma característica del filtro gaussiano empleado para el sensor y el ruido introducido para evitar el *inverse crime*.

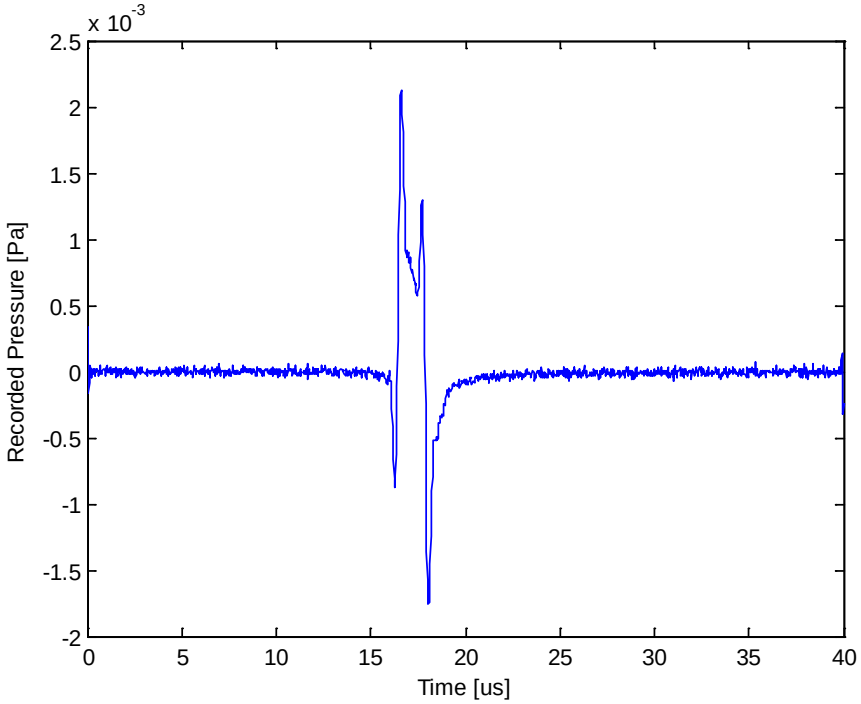

Figura 16: Gráfica Presión-Tiempo de la señal simulada con una posición

En la última gráfica se muestran los resultados de la reconstrucción donde se puede percibir ligeramente los bordes del tubo y la onda circular que sale de ellos y llega al sensor. Los valores de presión son mucho más bajos teniendo un máximo en torno a 0.0015 Pa mientras se estaba emitiendo 1 Pa. Esto puede ser debido a que las características del transductor empleado no son las idóneas para captar la señal emitida.

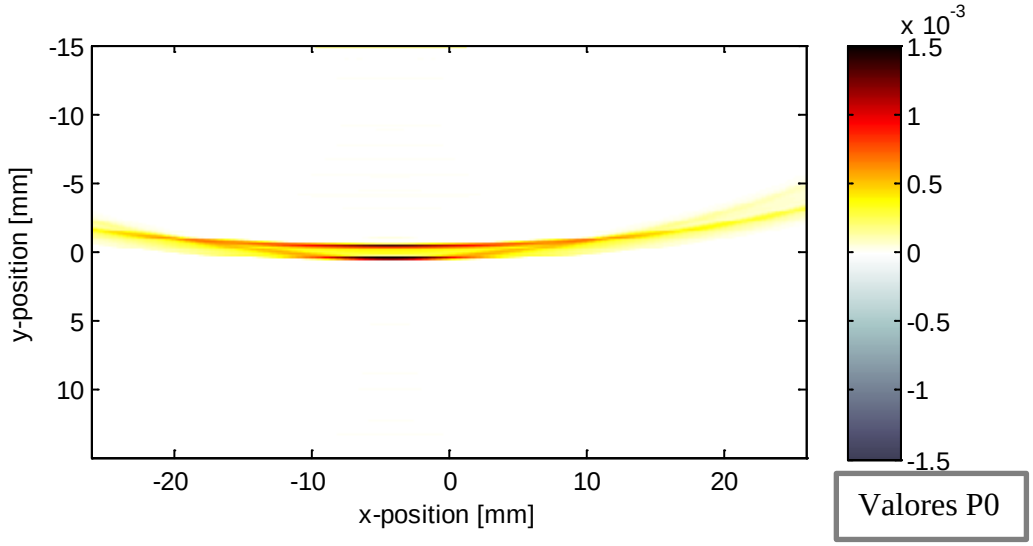

Figura 17: Gráfica de resultados de reconstrucción con una posición

3.2.3. Sensor en múltiples posiciones

El otro caso interesante para la simulación consiste en colocar el sensor en 4 posiciones diferentes a lo largo del phantom y recoger las 4 señales emitiendo la presión inicial el mismo número de veces. Este caso se repetirá el experimento en 4 posiciones diferentes para simular el uso de un *phased array* de 4 elementos y ver si existe mejora en los resultados. Para esta simulación se emplea la misma malla y la misma distribución de presión inicial que en la simulación anterior. Tampoco varían las propiedades del medio y los parámetros opcionales, así como el array de tiempo.

En el caso del sensor se ha ido modificando según la posición en la que se va situando. Para ello se ha colocado en primer lugar desde en la primer posición desde el punto 1 al 54, en la segunda del punto 55 al 108, en la tercera del punto 109 al 162 y en la cuarta del punto 163 al 216. Tras asignar el sensor se ha ido realizando la simulación en cada posición para así obtener señales diferentes en cada una.

```
% Sensor lineal binario Posición 1
sensor.mask = zeros(Nx, Ny);
sensor.mask(1,1:54) = 1;

% Simulación Posición 1
sensor_data1 = kspaceFirstOrder2D(kgrid, medium, source, ...
sensor, input_args{:});

% Sensor lineal binario Posición 2
sensor.mask = zeros(Nx, Ny);
sensor.mask(1,55:108) = 1;

% Simulación Posición 2
sensor_data2 = kspaceFirstOrder2D(kgrid, medium, source, ...
sensor, input_args{:});

% Sensor lineal binario Posición 3
sensor.mask = zeros(Nx, Ny);
sensor.mask(1,109:162) = 1;

% Simulación Posición 3
sensor_data3 = kspaceFirstOrder2D(kgrid, medium, source, ...
sensor, input_args{:});

% Sensor lineal binario Posición 4
sensor.mask = zeros(Nx, Ny);
sensor.mask(1,163:216) = 1;

% Simulación Posición 4
sensor_data4 = kspaceFirstOrder2D(kgrid, medium, source, ...
sensor, input_args{:});
```

A continuación, se han tratado esos datos para de nuevo obtener las medias de cada posición dejando una única señal por posición para simular como anteriormente se ha explicado que los múltiples puntos del sensor son un único sensor.

```
% Realizar la media de las señales obtenidas en el sensor
sensor_pres1 = sum(sensor_data1,1)/sum(sensor.mask(:));
sensor_pres2 = sum(sensor_data2,1)/sum(sensor.mask(:));
sensor_pres3 = sum(sensor_data3,1)/sum(sensor.mask(:));
sensor_pres4 = sum(sensor_data4,1)/sum(sensor.mask(:));
```

De nuevo se vuelve a emplear una malla distinta y a añadir ruido a las señales obtenidas para evitar el *inverse crime*. Cada señal es tratada por separado.

```
% Creación de otra malla
Nx = 256;           % Nº de puntos en x (Fila)
Ny = 256;           % Nº de puntos en y (Columna)
dx = x/Nx;          % Espaciado de puntos en x [m]
dy = y/Ny;          % Espaciado de puntos en y [m]
kgrid_recon = makeGrid(Nx, dx, Ny, dy);

% Añadir ruido a los datos obtenidos
signal_to_noise_ratio = 40; % [dB]
sensor_pres1 = addNoise(sensor_pres1, signal_to_noise_ratio, 'peak');
sensor_pres2 = addNoise(sensor_pres2, signal_to_noise_ratio, 'peak');
sensor_pres3 = addNoise(sensor_pres3, signal_to_noise_ratio, 'peak');
sensor_pres4 = addNoise(sensor_pres4, signal_to_noise_ratio, 'peak');
```

En cuanto a la reconstrucción, al haber empleado una malla diferente se adaptan las señales y la máscara del sensor a las nuevas dimensiones. Al haber aumentado la malla de 216 a 256 se aumenta en 10 puntos cada una de las posiciones, por lo que se comienza repitiendo la señal media a los 64 puntos de cada posición del sensor. Tras esto, se concatenan las 4 señales para asignárselas a la posición adecuada del sensor en la señal de reconstrucción final.

```
% Repetir la media para cada punto del sensor
sensor_pos1 = repmat(sensor_pres1,64,1);
sensor_pos2 = repmat(sensor_pres2,64,1);
sensor_pos3 = repmat(sensor_pres3,64,1);
sensor_pos4 = repmat(sensor_pres4,64,1);

% Unión de las señales repetidas para reconstruir
sensor_recon = cat(1,sensor_pos1,sensor_pos2,...
sensor_pos3,sensor_pos4);
```

Se crea la máscara del sensor para la reconstrucción, haciendo la unión de las cuatro posiciones ocupando todo el eje X. Esto es posible realizarlo de esta manera, ya que anteriormente se ha asignado la señal adecuada a cada posición y se han unido formando una única matriz. Se asigna el array de tiempo, la presión inicial a cero y se realiza la reconstrucción con los datos adecuados.

```
% Sensor lineal para la reconstrucción
sensor_rec.mask = zeros(Nx, Ny);
sensor_rec.mask(1,1:256) = 1;

% Asignar array de tiempo original
kgrid_recon.t_array = kgrid.t_array;

% Presión inicial a cero
source.p0 = 0;

% Asignar datos para reconstrucción
sensor_rec.time_reversal_boundary_data = sensor_recon;

% Realización de la reconstrucción Time Reversal
p0_recon = kspaceFirstOrder2D(kgrid_recon, medium, source, ...
sensor_rec, input_args{:});
```

A continuación, se muestra la primera de las gráficas para analizar los resultados de esta simulación. Se puede observar el tubo de presión inicial y la máscara del sensor con las 4 posiciones juntas que ocupan todo el eje X.

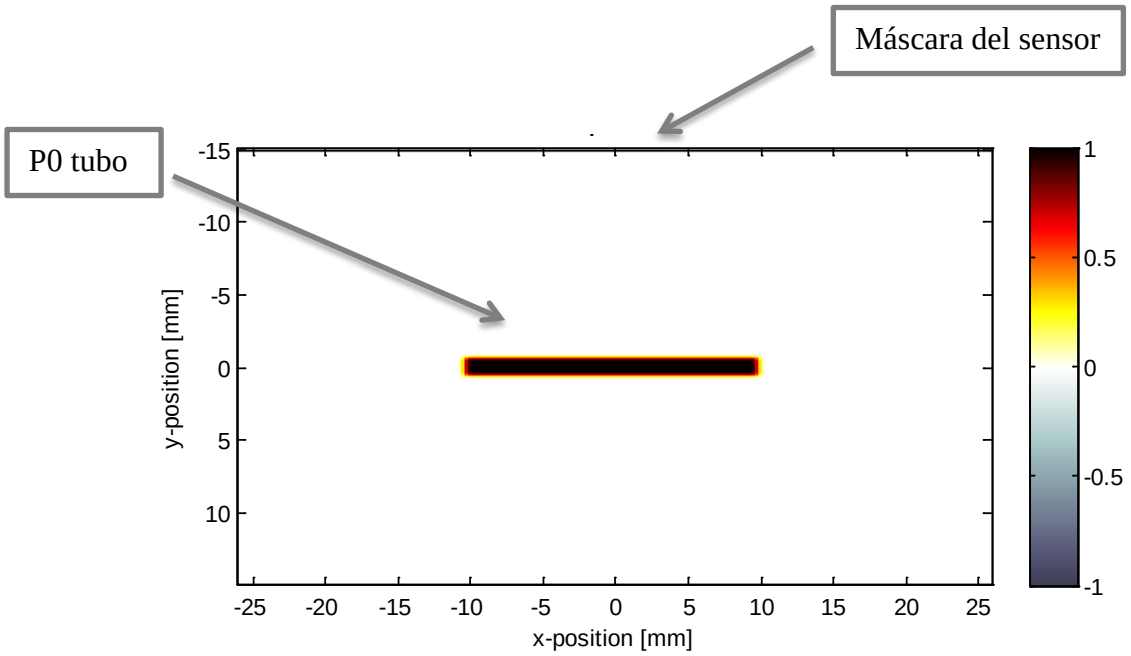

Figura 18: Situación inicial de reconstrucción con varias posiciones

La siguiente gráfica muestra la señal en presión-tiempo en cada una de las posiciones donde se ha colocado el sensor. Se puede observar que las posiciones centrales 2 y 3 y las exteriores 1 y 2 son muy similares entre sí debido a que la distancia a la presión inicial es también parecida. En las posiciones centrales se obtiene una señal mayor al estar más próximo al tubo. También se observa la forma gaussiana y el ruido añadido.

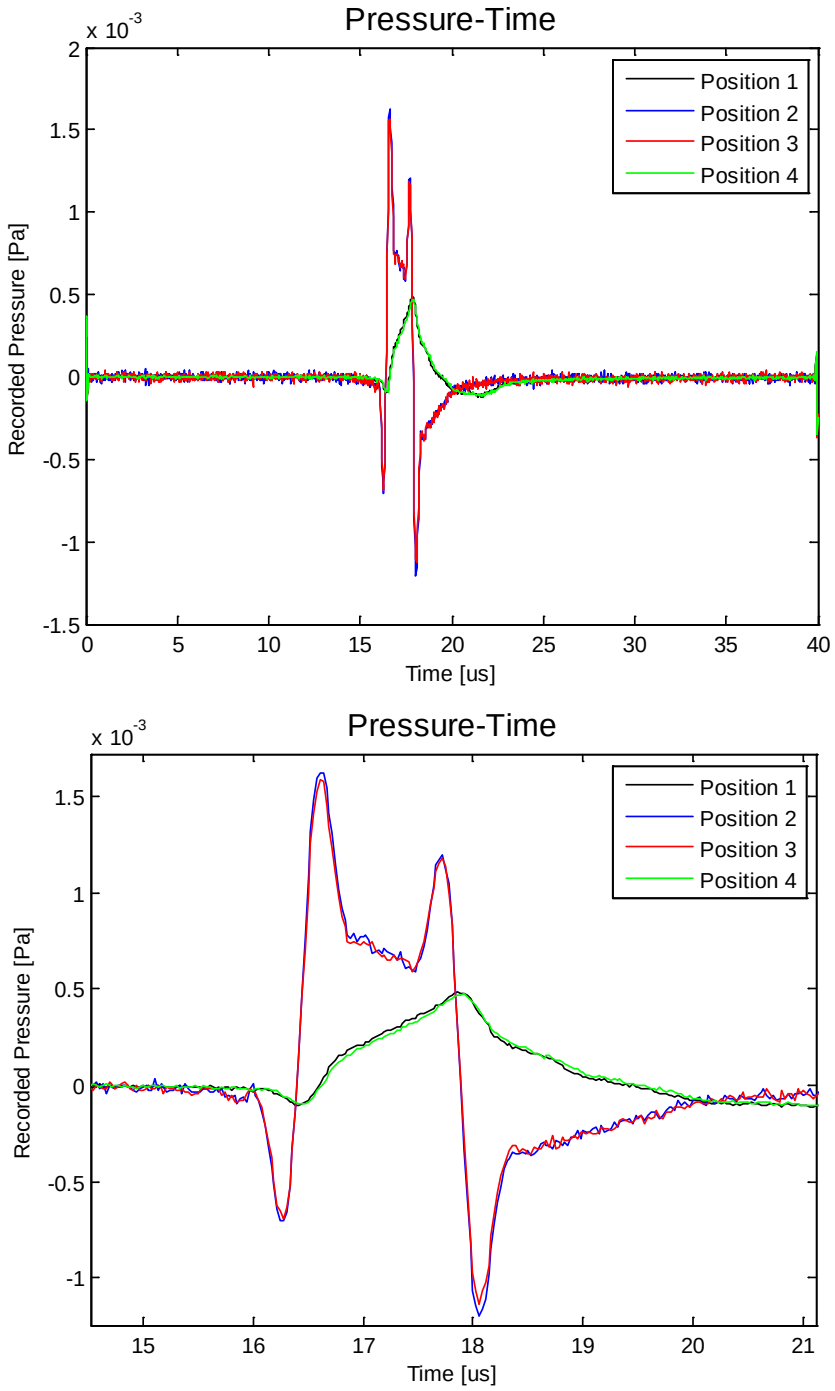

Figura 19: Gráficas de Presión-Tiempo de la señal simulada en varias posiciones

Por último, se muestra la gráfica de reconstrucción en la que se percibe una mejora en la forma del tubo con respecto a una posición, ya que el ancho y el largo tienen un mayor parecido a la distribución de presión inicial emitida pero los valores de presión siguen siendo bajos teniendo un máximo en torno a 0.0017 Pa mientras se estaba emitiendo 1 Pa.

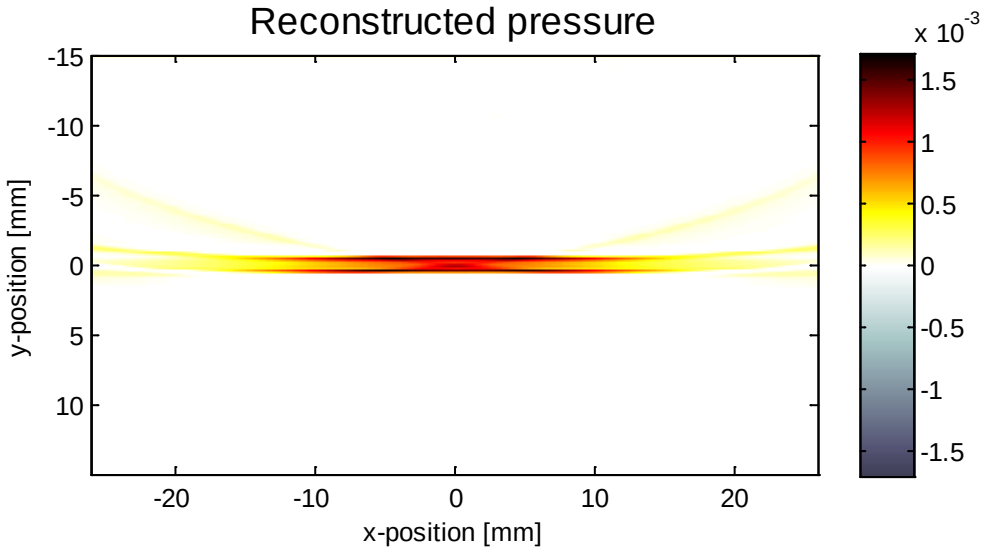

Figura 20: Gráfica de resultados de reconstrucción con varias posiciones

4. Conclusiones

En este trabajo se ha estudiado el efecto fotoacústico por medio de la realización de simulaciones adaptadas a condiciones reales. Este estudio es el primer paso hacia una futura investigación a realizar con materiales ya disponibles en el laboratorio. Los resultados obtenidos indican que existen mejoras cuando se emplean más posiciones de sensor combinando las distintas señales obtenidas. Pero también hay ciertos aspectos que se podrían tener en cuenta en futuros estudios.

Uno de los aspectos interesantes a estudiar, sería averiguar cuál sería el transductor idóneo para estas simulaciones. Para visualizar el problema se han obtenido las gráficas del espectro de amplitud de las señales emitidas en la simulación.

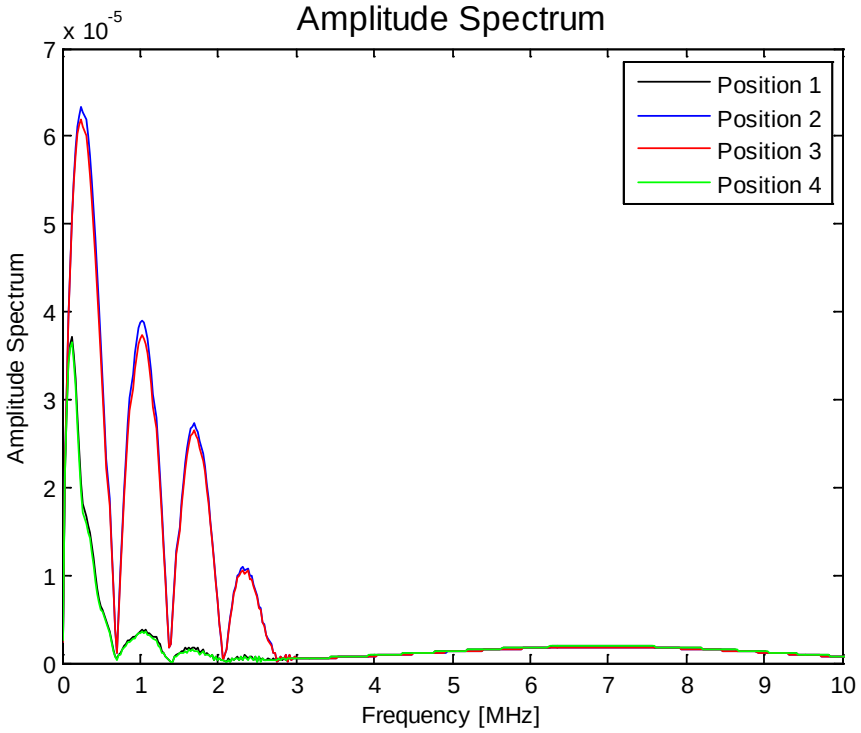

Figura 21: Espectro de amplitud de las señales recibidas con varias posiciones

Se puede deducir que la mayor parte del espectro de las señales emitidas se concentra entre los 0 y 3 MHz. El sensor utilizado, se había comentado, que concentraba su ancho de banda entre los 5 y los 10 MHz, lo cual puede influir en los resultados de la simulación. La herramienta *k-Wave* basa el cálculo de la frecuencia máxima soportada en la simulación empleando el espaciado de los puntos de la malla y la velocidad del sonido según la siguiente ecuación $f_{max} = \frac{sound\_speed}{2*dx}$. Como posibles soluciones podría tratarse de emplear otro sensor o cambiar la malla utilizada en busca de mejores resultados.

Otro de los aspectos interesantes a estudiar sería introducir los posibles valores de absorción en el medio y emplear un medio heterogéneo para ver su influencia sobre los resultados.

5. Bibliografía

[1] E. Marín, "*Escuchando la luz: breve historia y aplicaciones del efecto fotoacústico*", Lat. Am. J. Phys. Educ., vol 2. No. 2, 2008.

[2] P. Beard, "Biomedical photoacoustic imaging," Interface Focus, vol. 1, no. 4, pp. 602-631, 2011.

[3] Christian Lutzweiler y Daniel Razansky, "*Optoacoustic Imaging and Tomography: Reconstruction Approaches and Outstanding Challenges in Image Performance and Quantification*", Sensors, vol. 13, pp. 7345-7384, 2013.

[4] B. E. Treeby and B. T. Cox, "*k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields", J. Biomed. Opt., vol. 15, no. 2, p. 021314, 2010.*

[5] Bradley Treeby, Ben Cox y Jiri Jaros,"*k-Wave User Manual"*, Version 1.0.1, Nov. 15, 2012.

[6] *B. E. Treeby and B. T. Cox, "k-Wave help files*", (http://www.k-wave.org/documentation)