PROCEEDINGS of the
23rd International Congress on Acoustics

9 to 13 September 2019 in Aachen, Germany

MATLAB-based simulation software as teaching aid for physical acoustics

Jorge PETROSINO[1];Lucas LANDINI[2], Georgina LIZASO[2], Antonio IanKURI[2], IaninaCANALIS[2]

[1,2]Universidad Nacional de Lanús, Argentina

ABSTRACT

This paper presents examples of simulated experiments that can be run in MATLAB or Octave as a demonstration for students, allowing them to explore and understand the basics of wave propagation. The use of the k-Wave toolbox is proposed. This open source add-on is capable of performing simple and efficient simulation of wave propagation in the time domain, as well as providing comprehensible real-time visualization of the process. It is mainly used to simulate ultrasonic waves in biological media; however, its reliability isn't diminished in applications where lower frequencies are involved. In this work, 2D acoustic signals in the audible range are used to explore a variety of phenomena related to physical acoustics, such as reflection, diffusion, diffraction, absorption, resonance. The placement of the acoustic elements in the medium for each numerical experiment is extracted from a BMP image file depicting sources, sensors and solid materials, encoded in different colors. The proposed method allows users possessing elementary knowledge of MATLAB / Octave code to interact with the k-Wave toolbox in a beginner-friendly fashion. Vast control over the simulation conditions can be exercised by simply drawing, cutting, pasting, moving or recoloring elements of the image file.

Keywords: Teaching aid, Acoustical education, Numerical simulation

1. INTRODUCTION

k-Wave is an open-source MATLAB toolbox (1) most commonly used to simulate the time-domain propagation of acoustic wave fields in one, two and three dimensions. It has great versatility due to its employment of advanced numerical models which can account for both linear and non-linear propagation, an arbitrary distribution of heterogeneous material parameters, and different acoustic absorption models.

There is a wide selection of numerical methods for solving the differential equations that govern the propagation of mechanical waves. This toolbox uses the k-space pseudospectral method (2), which calculates the spatial derivatives through the Fourier collocation spectral method (3) and combines them with a temporal propagator expressed in the spatial frequency domain, or k-space.

The proposed method, unlike the finite difference and finite element methods, requires the use of a grid of equally spaced points that represent the area where the waves propagate. Since the spatial gradients are calculated using the FFT, the resulting wave field is implied to be periodic. This causes the waves propagating to one side and out of the bounds of the analysis field to reappear from the opposite side. To prevent this behaviour k-Wave adds a Perfectly Matched Layer, a thin layer that encloses the computational domain and provides absorption without reflecting the waves back into the medium. The source code, installation manual, publications and other useful information are available at *www.k-wave.org*.

One of the methods that the toolbox resorts to when defining the various parameters involved in the simulation process (medium density, wave propagation speed, source and sensor placement, among others) is the creation of matrices whose dimensions match the ones assigned to the base computational grid (also called *kgrid*). By doing so, the program separates the simulated environment into a layered structure, where one grid contains general specifications like the size of the analysis field and its resolution, another represents the type and placement of the acoustic sources, a third one indicates the position of sensors capable of recording time-dependent variables, and additional grids

[1]jorgepetrosino@gmail.com

can be added to specify the medium density at each point. Figure 1 depicts a breakdown of this operating framework.

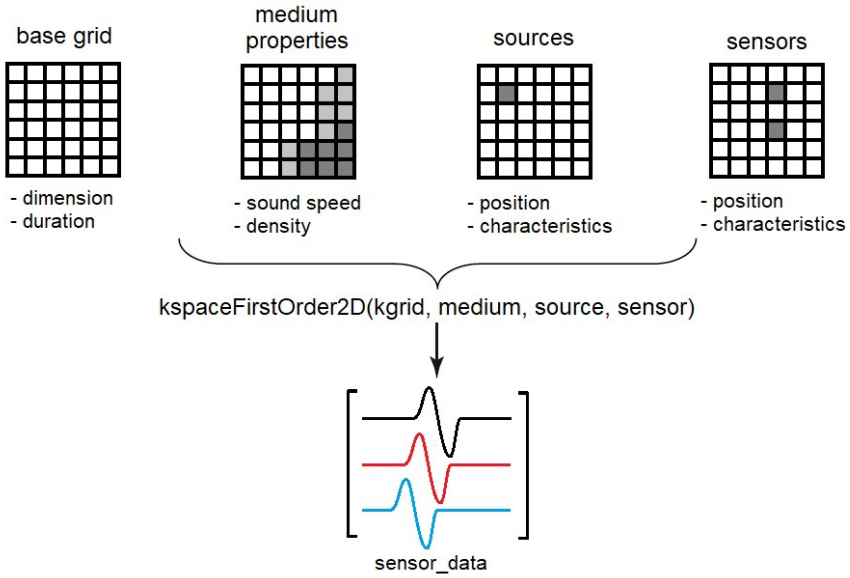

Figure 1 –Overview of the inputs structures and the main input fields used in k-Wave

The *kspaceFirstOrder2D* function, included in k-Wave, returns an array with as many rows as sensors were placed and as many columns as time samples were defined by the user. Its inner workings could be summarized in successive steps.

k-Wave has seen the most use in biomedical imaging research related to mechanical wave propagation in heterogeneous media (4,5), nevertheless, it is possible and relatively easy to adapt it to work with waves in the audible range propagating through the air.

The hereby presented proposal describes the use of a custom-made function specifically developed to allow students with little to no MATLAB coding experience to simulate a plethora of acoustic situations with this toolbox.

2. ADAPTATION FOR EDUCATIONAL USE

The aforementioned function enables the user to design the simulation structure through modification of the graphical properties of an image, letting him define the type of elements to be used by simply colour-coding them according to a set of pre-programmed references. The white areas of an image represent a homogeneous medium (air, by default) in which time-variant acoustic pressure sources can be placed arbitrarily, as well as fully reflective surfaces to serve as obstacles and sensors capable of recording the pressure variations over time. Figure 2a shows a simulation template example that uses a plane wave source (red line), a semi-circular array of 37 sensors with a 5º separation between elements, and a Schroeder diffuser.

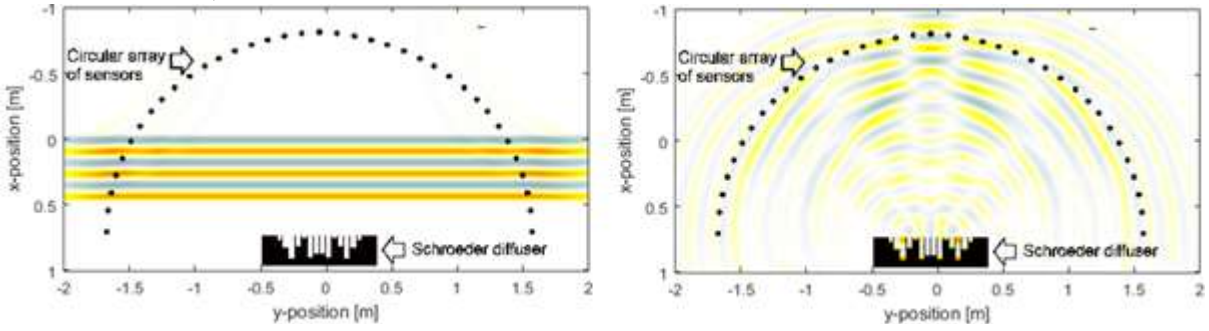

Figure 2 –Simulation of a plane wave's interaction with a Schroeder diffuser

simulateImage256, as the new function was named, reads the image file, builds the required input structures based on the information it provides and executes the simulation while simultaneously showing the corresponding animation on the screen. The default output variables of this function are

the temporal registers of the sensors, but the visual feedback can additionally be recorded on a video file if the user so chooses. Any sector of the image containing information encoded in grey colour is ignored by the simulator, which allows the inclusion of notes or other visual aids that can facilitate the understanding of the process (as seen in Figure 2).

Figure 2 illustrates the spatial distribution of the reflected wavefront. The simulation data provides the sound pressure levels registered by the 37 sensors over time. Figure 3b shows the sensor data from 3 of the measured positions (0º; 45º y 90º).

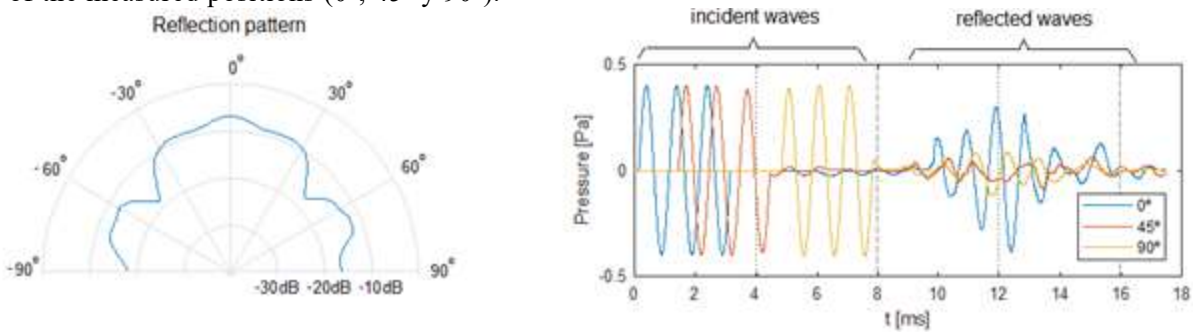

Figure 3 –Reflection pattern of the Schroeder diffuser (left), and measurements from three sensors located at 0º, 45º and 90º (rigth)

Each sensor first registers the passage of the plane wavefront and then the reflected wavefront. The signal energy values observed after the 9-millisecond mark correspond to the latter. Figure3a shows the diffuser's reflection pattern as registered by the complete sensor array, which can be plotted by calculating the energy levels of the signals after the passage of the incident wavefront.

With this simulation tool, a beginner student can try different configurations by simply modifying the image with most graphic editors. After altering its contents and saving the changes, just running *simulateImage256* will yield the results of a new simulation. In the examples presented here, the widely available and beginner-friendly Microsoft Paint was the editor software of choice, in order to minimize the learning curve. On the other hand, intermediate or advanced students still retain the option of directly modifying the code of the *simulateImage256* function to incorporate more complex characteristics to the medium before running the simulation.

3. DESCRIPTION OF THE PROPOSED FUNCTION'S MECHANICS

The input arguments the function receives are the name of the image file, a scale factor (side of the minimum square of the grid, expressed in meters), the simulation duration in seconds, a Boolean variable indicating whether the user wants to record the simulation to a video file or not, a sound speed value (m/s), and a structure array containing the acoustic source properties.

256-colour bitmap images must be used used. Red pixels (value 79) are interpreted as sources, green pixels (value 113) represent sensors and black pixels (value 0) correspond to perfectly reflective elements. It is possible to use multiple sources and sensors simultaneously without it having a negative impact on the total computation time.

The input variable source is, as mentioned before, a structure array that contains several fields: type, amplitude, frequency, number of cycles and duration. The most straightforward way to specify the temporal development of sources is to directly input the expression to be evaluated, using *t* as the variable that represents time. This is done by equating the *source.mode* variable to a string of characters containing the previously mentioned expression in text form. Eq. (1) shows an example of the generation of a source that emits the sum of two sine waves of different frequencies.

$$source.mode = '3*sin(2*pi*2000*t) + 10*sin(2*pi*1000*t)'; \quad (1)$$

Three pre-made source types that the user can set by entering their denomination have also been implemented. These correspond to an impulse, white noise and a burst of n cycles of a specified frequency. Each of them requires introducing additional parameters to various variables of the source structure. A pulse source requires specifying its amplitude by assigning a value to the variable *source.amplitude*, as shown in Eq. (2)

$$source.mode = 'impulse'; source.amplitude=2; \quad (2)$$

When using white noise it is necessary to enter the amplitude and duration of the noise, which must be less than the simulation's total duration (in seconds). This design choice responds to the need to see what happens after turning off the source, as shown in Eq. (3).

$$source.mode = 'whiteNoise'; source.span=1e\text{-}4; source.amplitude=2; \quad (3)$$

Lastly, the declaration of an *nCycles* source must account for the presence of a frequency value and number of cycles value, as shown in Eq. (4).

$$source.mode = 'nCycles'; source.amplitude=2; source.n = 3; source.f0=4000; \quad (4)$$

Aside from their specific requirements, all source types contain the variable *source.mode*, which can have two possible values: 'additive' or 'dirichlet'. The former injects the pressure as a mass source, while the latter enforces it as a dirichlet boundary condition. In certain types of experiments, it may be necessary to use one or the other. An important matter to keep in mind is that if during the course of its propagation a wave is faced with a dirichlet-type source, reflections will occur. It is recommended to use the default additive mode for simulations intended for educational purposes, reserving the dirichet mode for specific cases where it could be desirable to compare the behaviour of both modes.

The k-wave toolbox allows the use of multiple sources, each having an independent temporal definition. In order to strive for an increased ease of use, the present adaptation retains the capability to work with multiple sources, but sets all of them to share the same temporal definition by default. This means, in other words, that all sources will be emitting in phase.

After executing the simulation, the function *simulateImage256* returns several variables:

- **sensorData**, which is an array with as many rows as sensors were placed and as many columns as time samples were defined by the user,
- **t**, which is a vector corresponding to the time variable,
- **dt**, which is the minimum interval of time discretization,
- **lx**, which is the length in meters of the x-axis of the simulated space,
- **ly**, which is the length in meters of the y-axis of the simulated space.

4. PROPAGATION OF WAVES INSIDE TUBES

As an example of the simplicity of use of the tool, the propagation of waves in tubes with different edge conditions was simulated.

4.1 Propagation in an Infinite Closed-ended Tube

Figure 4a shows an instant of the post-reflection simulation at the closed end. The warm colours represent compression levels, and the cold colours rarefaction (positive and negative sound pressure respectively). Figure 4b is a plot of the temporal development data captured by the two sensors shown above. The blue line corresponds to sensor number 1, placed at the centre of the image, while the red line corresponds to sensor number 2, located next to the closed end.

It can be observed that the pressure wave is reflected without inversion at the closed end. It can also be noted that the instantaneous pressure level in sensor 2 is greater, since at the region near the closed end there is superposition between the progressive and regressive wavefronts.

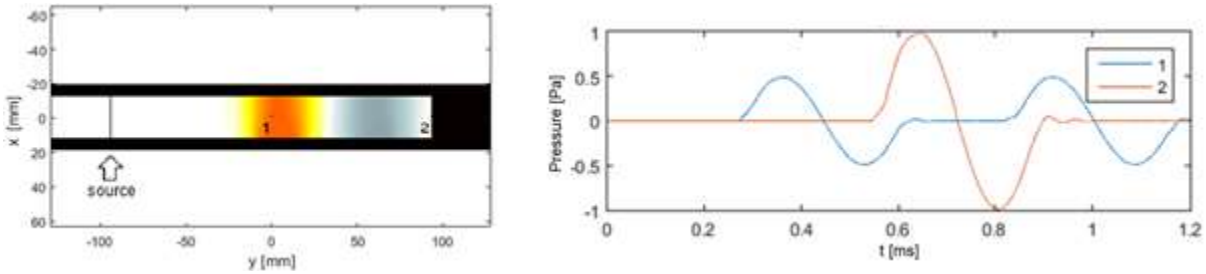

Figure 4 –Wave propagation in a tube with a closed end (left) and pressure at sensors (right)

4.2 Propagation in an Open-ended Tube

Figure 5 shows the simulation of an infinite tube that opens up at the region where the reflection takes place. In this example, one can notice the phase difference between the waves propagating to the left inside and outside of the tube.

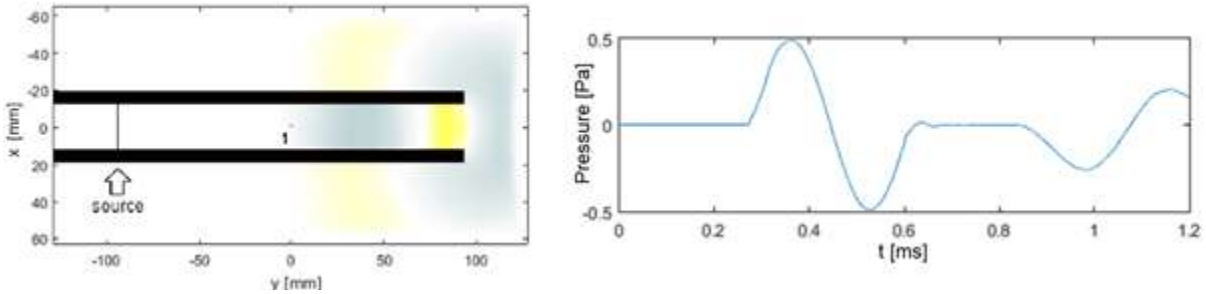

Figure 5 –Wave propagation in a tube with an open end (left) and pressure at sensor (right)

Observation of the data recorded by sensor 1 reveals that the reflected pressure wave is indeed inverted (Figure 5b). It can also be verified that the reflected wave has lower amplitude than the incident wave, due to the energy that is transmitted to the outside of the tube through the open end.

4.3 Results Comparison using the Finite Element Method

With the objective of comparing these results with the ones produced by more commonplace tools in the acoustics field, similar situations were simulated in COMSOL Multiphysics, which employs the finite element method. Figure 6 showcases the results of simulating an infinite tube with one of its ends closed. The blue stroke belongs to an acoustic pressure source and the green one to a sensor placed inside the tube, near the closed end.

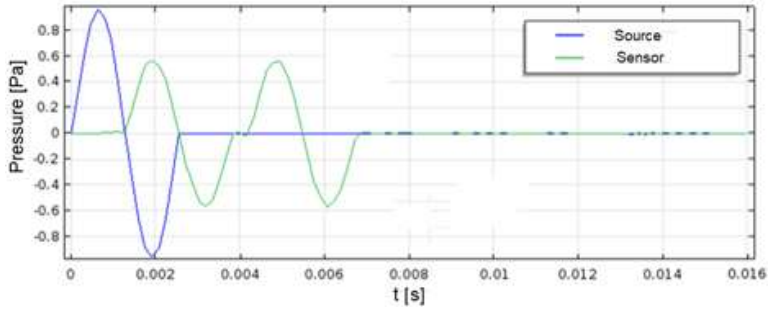

Figure 6 –Results of the finite element simulation of the reflection at the closed end

When using COMSOL, the simulation of a tube with an open end can be approached in several different ways. Figure 6 shows the results corresponding to a model where a zero-pressure boundary condition was enforced at the edge of the open end. In this case, it can be noted that the wave reflection is inverted but it lacks the energy loss observed in the previous open-ended example. This solution is consistent with the requested boundary condition, but it does not accurately represent the physical reality of wave propagation towards an opening. Results of the k-Wave simulation for the same tube are also overlaid in the following figure, showing a more adequate representation of this physical phenomenon.

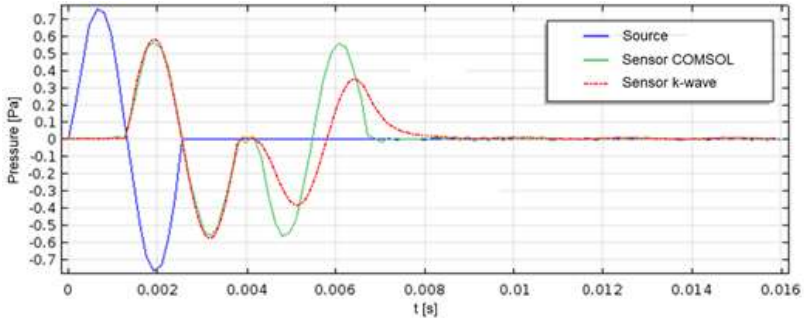

Figure 7 –COMSOL results when adding a zero-pressure boundary condition to the opening of the tube, compared to their k-Wave counterparts

Figure 8 illustrates the change in the COMSOL results after affixing an air chamber to the open end of the tube. This creates a more appropriate comparison, considering that the presented k-Wave examples take the external medium properties into account.

In this version of the simulation the reflected wave remains inverted, and the drop of amplitude levels associated with energy radiation to the exterior are also present. Different air chamber sizes would yield different results, but all of them closer to what could be measured in an experimental situation than the first attempt.

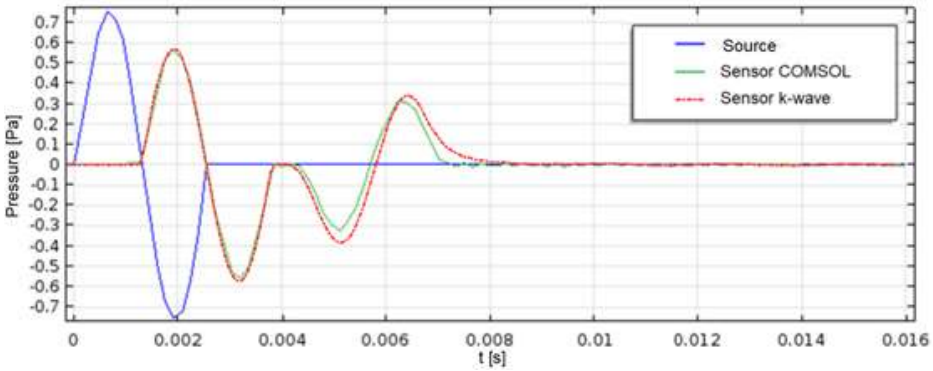

Figure 8 –Results after adding an air chamber to the open end of the tube in the COMSOL simulation, showing similarity with the k-Wave example

4.4 Sample code from tube simulations

The necessary code to perform a tube simulation like the ones previously reported requires few lines and is relatively easy to understand. Figure 9 provides an example of a MATLAB script capable of simulating the open-ended tube case.

```
infiniteOpenTube.m*
% Sample simulation of an infinite - open end tube

imageFileName = 'infOpenTube.bmp';
scale = 1e-3;
duration = 1.2e-3;
recordVideo = false;
c0 = 344;
source.type = 'nCycles';
source.amplitude = 0.5;
source.f0 = 3000;
source.n = 1;
source.mode = 'additive';

[sensorData, t, dt, equation, lx, ly] = simulateImage256(imageFileName, ...
scale, duration, recordVideo, c0, source);
```

```
openTubePlot.m*
% Plot of the simulation results

plot(t, sensorData);
xlabel('t [s]')
ylabel('pressure [Pa]')
legend('1','2')
```

Figure 9 –Sample code for the simulation of an infinite tube with an open end

The code structure remains the same for any simulation. Modifications can be made by simply editing the image or replacing it altogether, but changes to the source type and mode must be carried out by assigning different values to the source.type and *source.mode* variables, respectively.

Any edition performed on the image file in regard to the tube's shape or the placement and quantity of the sources and sensors will automatically provide the function with the information needed to execute the updated simulation, as long as the correct file name is passed to it.

5. TWO-DIMENSIONAL HEAD RELATED IMPULSE RESPONSE

The k-space pseudospectral method used by k-Wave to solve differential equations has no spatial numerical dispersion, and is able to correct the temporal numerical dispersion of the propagation through a homogeneous medium (1). This allows an impulse to be used as an excitation source and thus, to obtain impulse responses.

Figure 10 shows a sample image used to simulate the impulse response of a two-dimensional head, possessing a 180 mm diameter and 30 mm long ear canals, with two sensors placed at the end of each one.

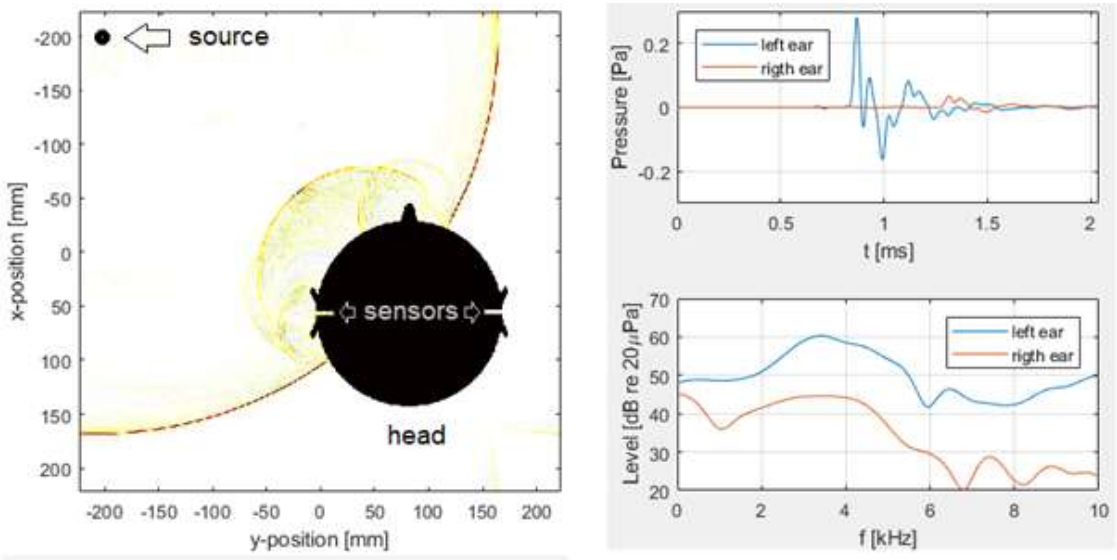

Figure 10 –Simulation for the recording of a 2D head-related impulse response

With the intention of comparing the alterations produced by the ear canals as well as the irregularities added to the base image, an analogous situation was simulated, in which the aforementioned ear canals, auricles and nose were removed, essentially turning the head model into a simple circle shape. In this case, the two sensors were placed on each side of the model, close to the perimeter of the circle (Figure 11).

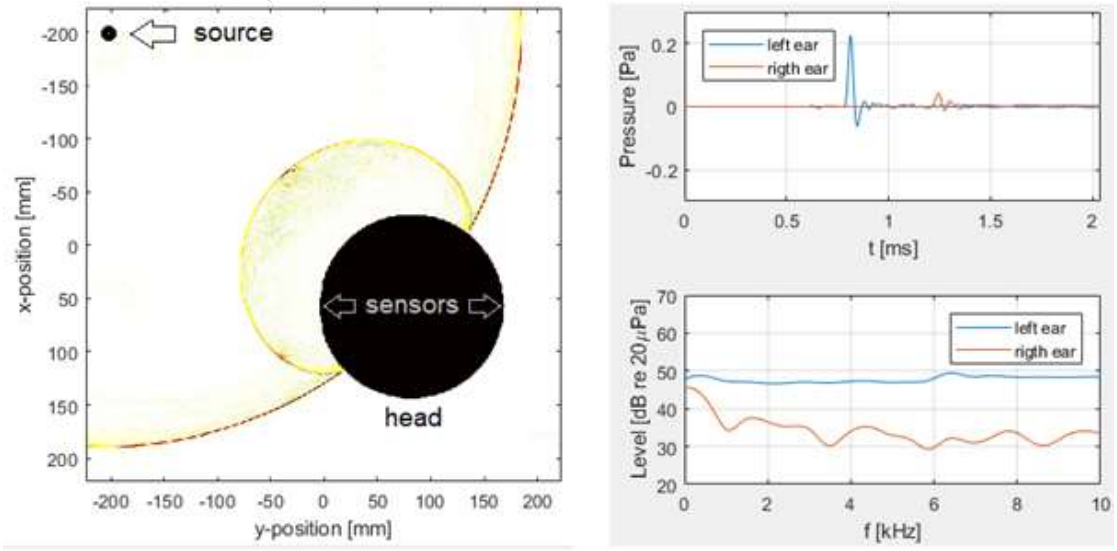

Figure 11 –Simulation with a simplified model (circle shape)

6. CONCLUSIONS

This work proposes the use of the k-Wave toolbox as a means of obtaining simulations useful for teaching physical acoustics. In an effort to simplify the workflow of said simulations a new function was incorporated, one that allows its users to define the structure and parameters of what they want to simulate by creating or editing image files. This could allow students, at least on a first approach to the subjects of choice, to concentrate on the theoretical matters without having to deal with the added complexity of manipulating the simulation algorithms.

Only a few examples of possible applications for this tool were presented. However, the added image editing capabilities combined with the fact that k-Wave has an open source code, give the user ample creative freedom for exploring various other phenomena such as interference, diffraction, diffusion, resonance and more.

The results obtained with this tool are consistent with the ones produced by finite element simulations, although it could be argued that software options involving that method have steeper learning curves and usually command higher prices, making them less suitable for beginner or

intermediate physical acoustics students.

The code for the *simulateImage256* function, along with examples and the image files needed to run them are available at (6).

ACKNOWLEDGEMENTS

The authors of this paper would like to thank the team integrated by Andrés Bonino Reta, Damian Andrés Fernández and NicolásCasaisDassie for their contributions during the investigation process.

The present work arose as a result of a research project currently under development at the National University of Lanús that requires the implementation of several wave propagation simulations assisted by k-Wave.

REFERENCES

1. Treeby BE, Cox BT. k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. J. of Biomedical Optics. 2010; 15(2):021314.
2. Tabei M, Mast TD, Waag RC. A k-space method for coupled first-order acoustic propagation equations. J Acoust Soc Am. 2012; 111(1):53-63.
3. Boyd JP. Chebyshev and Fourier spectral methods. 2nd ed. New York. USA: Courier Corporation; 2001.
4. Cox BT, Kara S, Arridge SR, Beard PC. k-space propagation models for acoustically heterogeneous media: Application to biomedical photoacoustics. J Acoust Soc Am. 2007; 121(6):3453-3464.
5. Cox BT, Beard PC. Modeling photoacoustic propagation in tissue using k-space techniques. In: Wang LV, Photoacoustic imaging and spectroscopy. 1st ed. Boca Ratón, USA: CRC Press, 2009.
6. Repository belonging to the user GLizaso at www.github.com, accessible through the following link: https://github.com/GLizaso/Teaching_aid_for_physical_acoustics.