MPHYGB27 MATLAB coursework 2017 – 2018

Due: Monday 15 Jan 2018, 11.55 pm

Erwin Alles, December 2017

Introduction and guidance

The tasks for the coursework are described in detail below. They are related to the simulation of ultrasound pressure fields generated firstly by idealised point sources and then by more realistic finite-sized sources as found in typical transducer arrays. Such simulations are essential for the development of novel imaging systems and algorithms, and can help to improve the image quality and functionality of clinical ultrasound scanners. All the required mathematical formulas and algorithms are provided, but you will need to think carefully about how to implement some of the tasks and which steps are involved.

When you submit your solution, you should:

- Submit all the .m files you created for this coursework. Include all functions and scripts.
- Name the functions and scripts in such a way that it is clear what each function or script does, e.g. `comp_press_field_point_source` and `Task1_script` for a function that computes the pressure field for a point source and for the script for Task 1, respectively.
- In addition, submit a text file that outlines which .m file corresponds to which task.
- Combine all scripts and functions into a single .zip file and upload this single file to Moodle.
- Include only your student number in the file name – do not include your name!

Please note:

- Do not submit graphics and figures; instead ensure your scripts generate the figures at run-time in exactly the way you wish them to look.
- In order to be considered, your solution to the coursework must be submitted via Moodle by 11.55pm on Monday 15 January 2018. Extensions to this deadline can only be permitted in cases where approval has been granted from the Medical Physics & Biomedical Engineering Department extenuating circumstances committee

This MATLAB coursework accounts for 34% of the total course mark. Marks for this coursework will be awarded using the following criteria:

- Up to 60% will be based on whether your code correctly implements the methods and mathematics described for each task, and in addition produces the correct results and clear visualisations.
- Up to 25% will be based on how easy it is to understand and use your code. This includes the use of both internal and external comments, informative variable and function names, and clarity and structure of the code.
- Up to 15% will be based on adhering to other good programming practices that you have learned about in the lectures, such as memory management, error checking, and vectorisation.

Marks for implementation correctness will be awarded for each task individually (the marks available are indicated next to each task number), whereas the marks for clarity and good practice will be awarded based on the whole of your solutions.

Please take careful notice of the UCL guidelines on plagiarism (https://www.ucl.ac.uk/students/exams-and-assessments/plagiarism), and be aware that submitted coursework will be compared for similarity with other submitted coursework and code available elsewhere (e.g. on the internet). This does not prevent you from discussing the coursework with others or using the various sources of help mentioned in the lectures, it just means you have to devise and write your own code.

If you have any questions concerning the tasks regarding *what* you need to do (as opposed to *how* to solve these tasks), please contact me through Moodle or directly by e-mail (e.alles@ucl.ac.uk).

Task 1 [10]

For the first task, you will be performing simulations that compute the pressure field generated by an acoustic point source (i.e., an acoustic source of infinitesimal dimensions that emits a δ-distribution shaped pressure waveform; you have seen the δ-function in the DSIP lectures). Mathematically, this pressure field is given by

$$p(x,y,z,t) = p_0 \frac{\delta(t - r/c)}{4\pi r}, \qquad r = \sqrt{(x-x_s)^2 + (y-y_s)^2 + (z-z_s)^2}, \tag{1}$$

where c is the speed of sound, (x,y,z) are the co-ordinates of the grid point where the pressure is to be evaluated (measured in m), (x_s,y_s,z_s) are the co-ordinates of the acoustical point source, r is the distance between each grid point and the point source, p_0 the initial pressure amplitude, and t the time (in seconds). Examining Equation (1), we see that the pressure amplitude generated by a point source is proportional to the inverse of distance r, and that the acoustic wave propagates spherically away from the source.

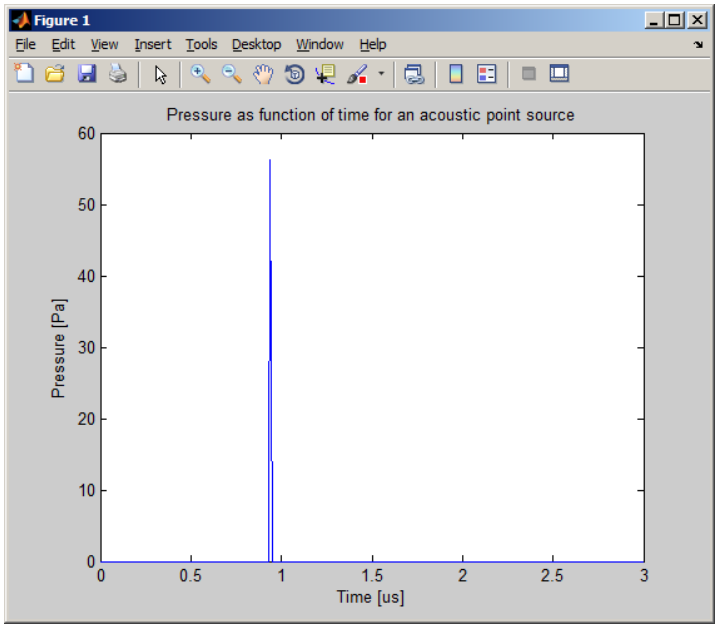
Write a function "`comp_press_field_point_source.m`" that computes the pressure as a function of time over a three-dimensional (3D) grid of sample points. This function should have the following inputs: speed of sound c; amplitude p_0, co-ordinate values for the three axes x, y and z (each given as a vector) that define the spatial grid point(s) where the pressure will be calculated; source coordinates x_s, y_s and z_s; and all time points for which the field is computed. This function should produce a four-dimensional array containing the pressure for every point in space and time.

Next, write a script that uses your function to compute the pressure at grid point (x,y,z) = (1,1,0) mm for all time samples $0 \le t \le 3$ µs using a temporal step size of Δt = 10 ns. Use the following parameters:

- $x_s = y_s = z_s = 0$ mm
- c = 1500 m/s to simulate water or soft biological tissue
- p_0 = 1 Pa·m (note: this constant has "unusual" units to ensure the computed pressure will be given in Pa!)

Your script should plot the resulting pressure against time. Make sure you set the horizontal axis to seconds or microseconds, and add informative axis labels and titles to this plot.

Finally, convince yourself your function is working correctly by comparing the arrival time of the acoustic pulse (i.e., the time it takes for an acoustic wave to propagate from the source to the grid point) and the pressure amplitude with values predicted using Equation (1).



Example output for Task 1

Hint: the distribution $\delta(t)$ has the properties $\delta(t \neq 0) = 0$ and $\int_{-\infty}^{\infty} \delta(t)\mathrm{d}t = 1$. These properties are equally true for the discretely sampled case: $\delta[t \neq 0] = 0$ and $\sum_t \delta[t] = 1$. Use these properties to work out how to implement a δ-distribution on a discrete grid.

Task 2 [10]

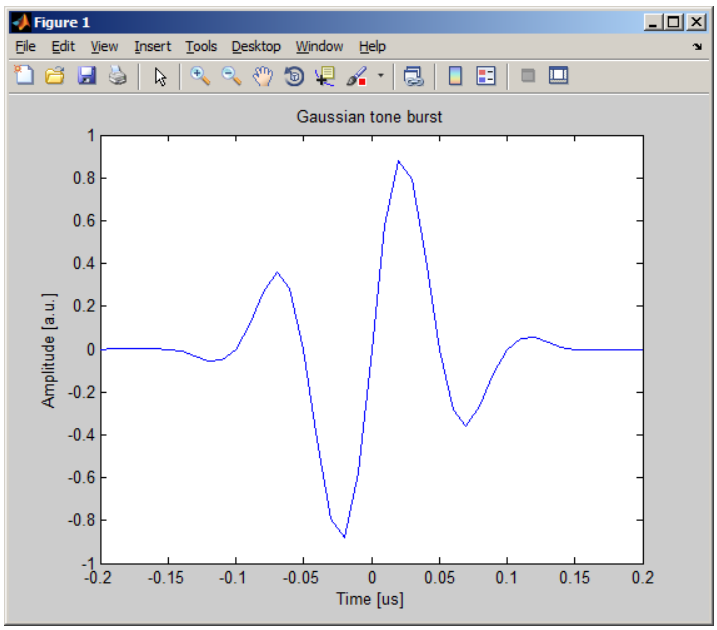
The situation you have considered in Task 1 is not very realistic; actual acoustic sources generate acoustic pulses that have a finite duration in time and contain frequency components over only a limited range of frequencies (i.e. the acoustic energy is mainly contained within a finite range of frequencies). In the next few tasks, you will extend the simulation by incorporating a more realistic acoustic pulse shape. The waveform we will consider is called a normalised Gaussian tone burst, *G*, and is given by the equation

$$G(t; f_0, \sigma) = \exp(-t^2/2\sigma^2)\ \sin(2\pi f_0 t), \tag{2}$$

where t is time (in seconds), f_0 is the centre frequency (in Hz), and σ is the standard deviation (in seconds) of the Gaussian envelope given by $\exp(-t^2/2\sigma^2)$. Here, standard deviation σ determines the width in time of the tone burst.

First, write a function "`comp_Gaussian_tone_burst.m`" that computes a Gaussian tone burst for arbitrary input values of centre frequency f_0, standard deviation σ, and temporal step size Δt. You have learned in the DSIP lectures that a Gaussian over the range -4σ to $+4\sigma$ is a good approximation to a Gaussian function; hence, have your function compute the Gaussian tone burst for $-4\sigma \le t \le +4\sigma$.

Next, write a script that uses this function to compute a Gaussian tone burst for the following parameters: f_0 = 10 MHz, σ = 50 ns, and Δt = 10 ns. Your script should plot the Gaussian tone burst against time as shown in the figure below. In addition, convince yourself that the output is correct. (Hint: $G(t; f_0, \sigma)$ is antisymmetric, hence $\int G(t; f_0, \sigma)\, \mathrm{d}t = 0$. Also, check whether the amplitude of $G(t; f_0, \sigma)$ is as you expected.)



Example output for Task 2

Task 3 [10]

The acoustic pressure given by equation (1) describes the *impulse response* of an idealised acoustic system where an acoustic point source emits a pulse of ultrasound. This pulse, which is described by the δ-distribution, is of infinitesimal duration in time and hence exhibits infinite bandwidth (i.e., all frequency components are equally represented within the signal).

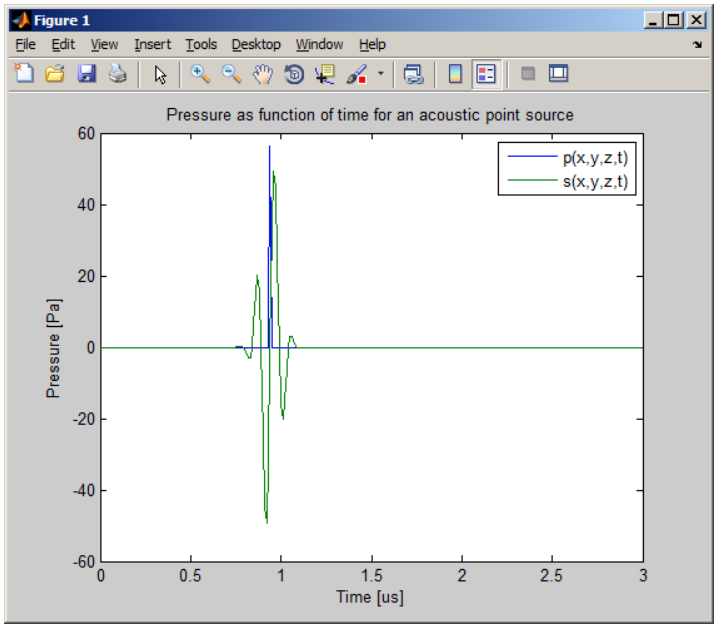
An acoustic pulse shape with a finite duration in time, such as the Gaussian tone burst $G(t; f_0, \sigma)$ of Task 2, exhibits a finite bandwidth and only contains frequency components across a finite range of frequencies. Thus, incorporating an acoustic pulse with a finite duration in time can be viewed as applying a linear filter (which you have seen in the DSIP lectures) that suppresses those frequency components not present in the pulse shape of finite duration in time. Consequently, the pressure generated by an excitation signal of finite temporal duration is computed by convolving the impulse response *p* and the excitation function *G*, i.e.,

$$s(x, y, z, t) = p(x, y, z, t) *_t G(t; f_0, \sigma), \tag{3}$$

where $*_t$ denotes the convolution operator performed over time.

In a new script, repeat Task 1 but in addition incorporate a Gaussian tone burst into your acoustic field simulations (hint: look up MATLAB's `conv` function). Use the same parameters as in Task 1 to compute *p*(*x*,*y*,*z*,*t*), simulate the output *G* of a Gaussian tone burst using the parameters listed in Task 2, and plot the resulting pressure against time – an example is shown in the figure below. Display both *p*(*x*,*y*,*z*,*t*) and *s*(*x*,*y*,*z*,*t*) in the same plot, make sure you reuse the functions you wrote for Tasks 1 and 2, and add informative axis labels, legends and title to the plot.

As before, convince yourself the results are correct through comparison of the arrival time (i.e., the time it takes for an acoustic wave to propagate from the source to the grid point), pulse width and amplitude against values predicted using Equations (1)-(3).



Example output for Task 3

Task 4 [15]

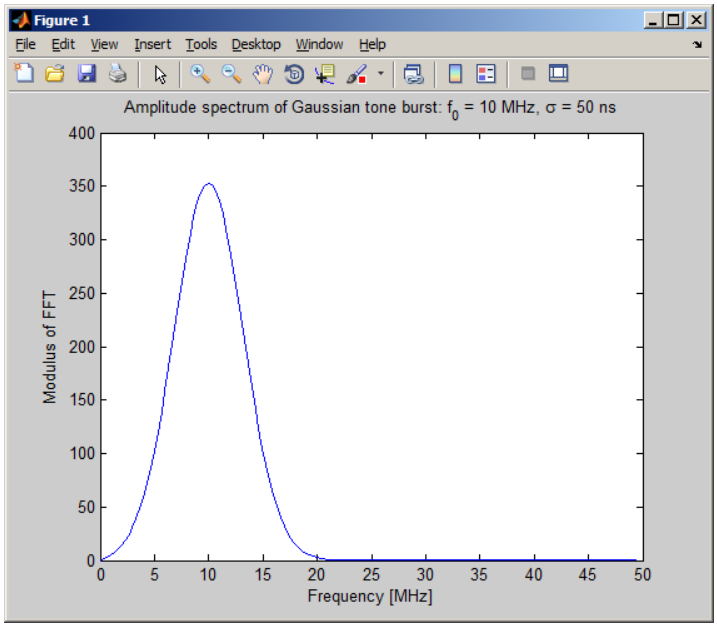
You have learned in the DSIP lectures that you can obtain an amplitude spectrum by computing the magnitude (also called modulus) of the (complex-valued) Fourier transform of excitation signal $s(t)$, i.e.,

$$S(f) = |F\{s(t)\}|.$$

Write a script that computes the amplitude spectrum $S(f)$ of $s(x,y,z,t)$ as computed in Task 3, and plot this spectrum as a function of frequency f. Next, vary the values of f_0 and σ, compute $s(x,y,z,t)$ for these new values, and plot the resulting amplitude spectrum in the same figure but in a different panel. Repeat this for six different pairs of values of f_0 and σ in total; the expected output is a single figure containing six separate plots using six different panels. Only show the amplitude spectra for positive frequencies, and, as before, add informative axis labels and titles to each panel. An example of one of these six panels is shown in the figure below.

Finally, using comments in your script, describe your observations: what happens to the amplitude spectrum if you increase or decrease f_0, and what happens when you increase or decrease σ? To obtain meaningful results, please limit the parameters to $2 \leq f_0 \leq 40$ MHz and $5 \leq \sigma \leq 500$ ns.

Note: when computing a discrete Fourier transform using the `fft` ***function, MATLAB returns first all the components corresponding to positive frequencies, followed by all components corresponding to negative frequencies. Hence, the frequency axis runs first from 0 Hz to the Nyquist frequency, followed by -Nyquist to 0.***



Example output for Task 4 for one set of parameters

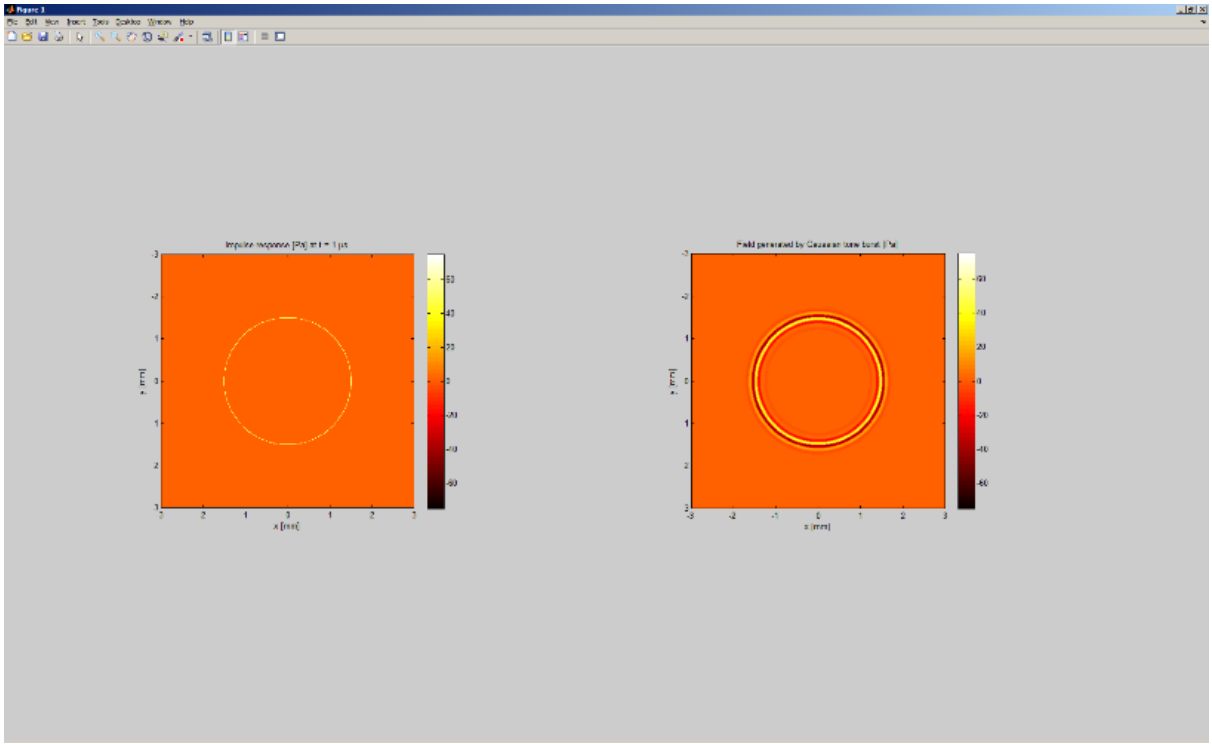
Task 5 [15]

Now write a script that computes the pressure generated by an acoustic point source across an extended xy-plane rather than in a single point. Use the following values for the various input arguments:

- Spatial grid:
  - -3 mm ≤ x ≤ +3 mm,
  - -3 mm ≤ y ≤ +3 mm,
  - z = 0 (i.e., we are only interested in the pressure field in the x-y plane),
  - Use a step size of 10 µm in every direction.
- Temporal axis: 0 ≤ t ≤ 3 µs, temporal step size of 10 ns
- Speed of sound as c = 1500 m/s, p_0 = 1 Pa·m
- Source location: (x_s, y_s, z_s) = (0, 0, 0)
- Gaussian tone burst parameters as in Task 2: f_0 = 10 MHz, σ = 50 ns, and Δt = 10 ns.

Reuse function `comp_press_field_point_source` (which you wrote for Task 1) to compute $p(x,y,z,t)$ for all points in space and time contained within the computational grid, and reuse function "`comp_Gaussian_tone_burst`" (which you wrote for Task 2) to implement a Gaussian tone burst.

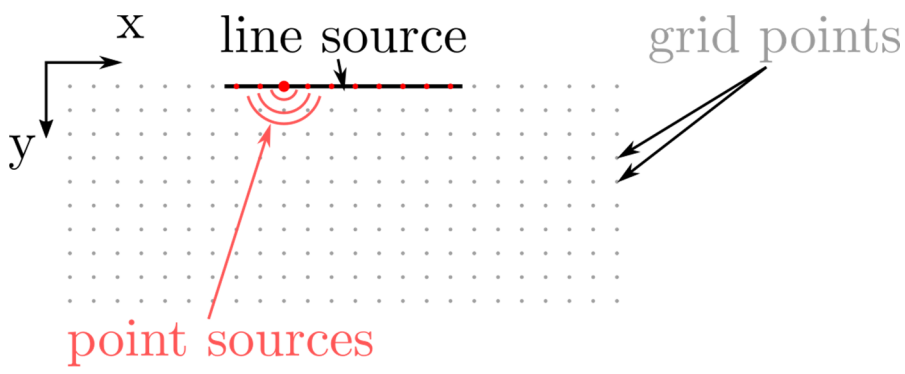
Your script should display the pressure field across the plane defined by x and y, as observed at t = 1 µs. In the same figure, show the impulse response p and the response to a Gaussian tone burst s in two separate panels (as shown below). Use the actual co-ordinates (in mm) along the axes, and ensure the pressure fields appear correctly (i.e., the pressure field exhibits the circular symmetry described by Equation (1)). Note that pressure p contains only positive values, whereas s contains both positive and negative values – please ensure that the amplitude scales in both panels range between the same limits. As usual, add meaningful labels and titles to the panels, and finally display a colour bar that explains which colour corresponds to which value.



Example output for Task 5

Task 6 [20]

Finally, you will extend this simulation to acoustic sources of finite dimensions. To achieve this, we make use of Huygen's principle (http://dosits.org/science/advanced-topics/how-does-sound-move-wave-propagation-and-huygens-principle/), which states that any wave front can be treated as an infinite collection of infinitesimally small point sources. Thus, we can approximate the field generated by an acoustic source of finite spatial extent as the sum of a finite collection of point sources, provided that these point sources are spaced sufficiently close to avoid (spatial) aliasing. In this task, you will model an acoustic line source located at ($|x| \leq 0.5$ mm, y = z = 0) as a collection of point sources. For simplicity, each of these point sources will coincide with a grid point, as is shown schematically below:



Write a script that performs the following actions:

- Determine and loop through all grid points located *within* the line source;
- Place a single point source in each of these grid points and compute the pressure fields for each of these individual sources in all points in space and time;
- Sum the pressure fields generated by the individual point sources to approximate the field generated by the acoustic line source of finite dimensions;
- Apply a Gaussian tone burst as the excitation function.
- Use the following parameters:
    - Speed of sound as c = 1500 m/s, p_0 = 1 Pa·m
    - Gaussian tone burst parameters as in Task 2: f_0 = 10 MHz, σ = 50 ns, and Δt = 10 ns.
- And evaluate the pressure field on a computational grid defined by
    - Spatial: -2 mm ≤ x ≤ +2 mm, 0 mm ≤ y ≤ +4 mm, z = 0
        - Use a step size of **50 µm** in every direction.
    - Temporal: 0 ≤ t ≤ 4 µs, using a temporal step size of Δt = 10 ns

The output of your script should be a figure displaying the total pressure field observed at t = 1 µs, and contain informative axis labels and title.

Notes:

- ***Computing the complete simulation might take a few minutes. During coding and debugging, it might be convenient to limit computations to only the first few source points to speed things up.***
- ***The temporal convolution required to apply a Gaussian tone burst is a linear operator, which means that*** $\boldsymbol{p_1(t) *_t G(t,\sigma) + p_2(t) *_t G(t,\sigma) = \left(p_1(t) + p_2(t)\right) *_t G(t,\sigma)}$***. Use this to your advantage to avoid repeated temporal convolutions.***
- ***Please check you use the correct parameters. If you keep using the old parameters (from Tasks 1-5), you may find that the computation can take orders of magnitude longer to complete!***

Task 7 [10] – N.B. M-level students only!

In this final task, you will measure the variation in the accuracy of the simulation performed for Task 6 as a function of the separation distance between the point sources. The accuracy of the simulated pressure field can be computed using the normalised root-mean-square (RMS) error *E* (in %),

$$E = 100 \cdot \frac{\sqrt{\mathrm{E}[\{s_{\mathrm{current}}(x,y,z,t) - s_{\mathrm{reference}}(x,y,z,t)\}^2]}}{\sqrt{\mathrm{E}\left[s^2_{\mathrm{reference}}(x,y,z,t)\right]}}, \tag{4}$$

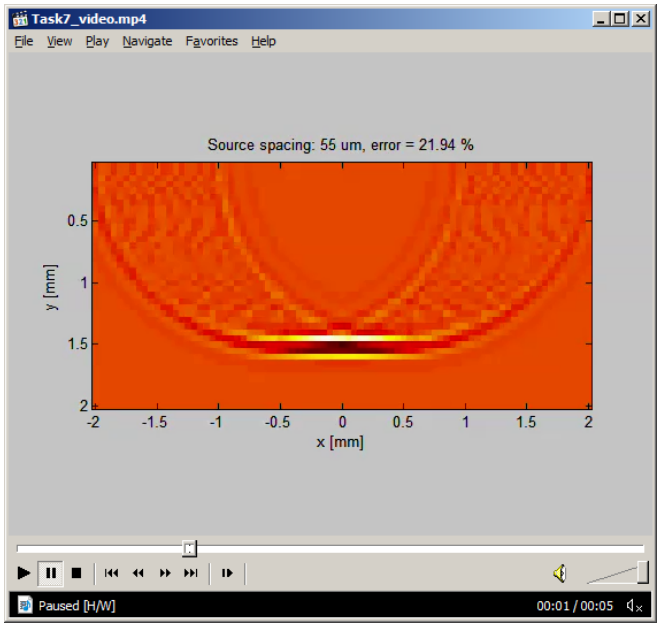
Where E[…] denotes computing the mean (or expected value) of the argument across all sample points in time and space. In order to evaluate Equation 4, we first need to compute a reference solution. As for this geometry (i.e., a line source of finite length) the closed-form analytic solution is rather complicated, you will compute a numerical solution instead that can be assumed to be (close to) exact.

Write a script that performs the following actions:

- Compute the reference solution by distributing a large number of point sources along the line source, at a source separation distance of 1 µm. This small source separation distance results in significant oversampling (i.e., the sampling density is much larger than that required to satisfy the Nyquist criterion) – this over-sampling technique is often used to obtain numerical reference solutions.
    - Use the parameters of Task 6, and include a Gaussian tone burst as excitation function
    - Evaluate the pressure field on a computational grid defined by
        - Spatial: -2 mm ≤ *x* ≤ +2 mm, **50 µm ≤ *y* ≤ +2 mm**, *z* = 0
            - Use a step size of **50 µm** in every direction.
            - **Note: the y-axis starts at 50 µm rather than at 0 mm to avoid dividing by zero in Equation (1)!**
        - Temporal: **0 ≤ *t* ≤ 3 µs**, using a temporal step size of Δt = 10 ns
- Next, compute the pressure fields generated by the line source using point source separation distances ranging from 5 µm to 200 µm, in increments of 5 µm. For each of these pressure fields (evaluated in the same spatial co-ordinates), compute the normalised RMS error.
- Generate an animation displaying the pressure field generated by the line source for all tested source separation distances, and include the reference solution. Display the pressure field at the time point *t* = 1 µs. Each frame of this animation should display the pressure field obtained using a different point source separation distance. Add informative axis labels.
- Display both the point source separation distance (in µm) and normalised RMS error (in %) in the title of the figure in each frame. Hint: look up the `title` and `sprintf` functions to work out how to update the title for each frame. An example frame is shown in the figure below.
- Finally, record all these frames into a video using mpeg-4 encoding and a frame rate of 8 frames per second (FPS).

Notes:

- ***Do not submit the generated videos; only submit the script as your solution!***
- ***Where possible, use*** `MovieWriter` ***for this task. However, if you do not have access to a recent version of MATLAB, you will have to use*** `movie2avi` ***to complete this task – in that case, do not apply video compression/encoding.***
- ***For different point source separation distances, you will have to use different numbers of point sources to fully populate the line source. Make sure you normalise the amplitude of the total pressure field obtained for each point source separation distance by dividing the total field by the number of point sources used – otherwise the pressure amplitude of the total field would depend on the point source separation distance.***



Example output for Task 7