4 – Huygens' Modelling of Transducers and Arrays

Objectives

The purpose of this exercise is to model the field from a transducer and ultrasonic array in 2D when the device transmits a continuous wave at a single frequency using Huygen's principle. This technique is a useful starting point for designing an array in the assessed coursework for the unit.

Summary

Huygens' principle is used to model the field from a transducer by splitting the transducer up into multiple point sources and superposing the simple fields from the individual sources. In this exercise a 2D model will be considered. This corresponds to a cross section through the field from a rectangular transducer that is long in the direction perpendicular to the plane of the model. In 2D, such a transducer is modelled as a single row of point sources that each emit cylindrical waves in the plane of the model. To model the field when a transducer transmits a pulse, the field for each frequency component in the pulse must be simulated separately and the results combined (similar to exercise 2). This will give the field at an instant in time, and in order to fully model the field, the result has to be recomputed at different instants in time as the pulse propagates away from the transducer. This is time consuming due to the need to perform the simulation over multiple frequencies and at different times. However, a quick way of estimating the maximum field intensity that will occur anywhere in space is to consider the transducer emitting a continuous wave at the centre frequency of the pulse. The same principle can also be used to model the maximum intensity of the field from an ultrasonic array when a particular focal law is applied.

Exercise 4A

Load the program `STARTER_huygens_a` and examine the code that has been written. The necessary input lines and code to plot the output have been given – all you need to do is to add the code for the calculation. Note that there are several ways of achieving the required output. The comments already in the program should provide help with the structure and these highlight the essential steps which are:

1. Set up vectors `x` and `y` for the coordinates of the output grid. `x` (lateral distance) should be from `-grid_size_x/2` to `grid_size_x/2` and `y` (distance from transducer) should be from `0` to `grid_size_y`. The step size in each should be approximately equal to `grid_pixel_size`.
2. Set up a vector `source_x_positions` of the x coordinates of the sources to represent the transducer. These should extend from `-transducer_width/2` to `transducer_width/2` and contain enough elements so that there are at least `min_sources_per_wavelength` sources per wavelength.
3. The output matrix, `p`, will hold the results.
4. Write the code to perform the Huygens' calculation.
5. When the program is working, examine the effect of changing the input parameters `min_sources_per_wavelength`, `transducer_width` etc..
6. Try to optimise the program for speed of execution. (e.g. `for…` loops are slow - it is much faster to use matrix operations as much as possible).

Exercise 4B

Load the program `STARTER_huygens_b.m`. This contains the start of a program to simulate the field from an ultrasonic array using Huygens' principle. In this context, Huygen's principle predicts the maximum field intensity when a given focal law is applied to the array. The initial parameters to use for the array transducer have already been coded and are: 32 elements; 5 MHz centre frequency; 0.6 mm element pitch; 0.5 mm element width. Your program should output three images in the same figure (the `subplot` function to do this is in the code) corresponding to the array being focused at the three focal points that have already been coded in the matrix `focal_positions`. These are: $(x, y) = (-10, 20)$ mm; $(0, 30)$ mm; $(10, 40)$ mm.

When the programme is working, adjust the parameters of the array (`no_elements`, `element_pitch`, `element_width` and `frequency`) and see how the results change. Can you draw any conclusions?

To account for the finite width, a, of the array elements (i.e. `element_width` in the code) when computing the field from an individual element, you should use the element directivity function provided in the notes and repeated here:

$$D(\theta) = a\,\mathrm{sinc}\left(\frac{1}{2}ka\sin\theta\right) = a\,\mathrm{sinc}\left(\frac{\pi a \sin\theta}{\lambda}\right),$$

where θ is the angle relative to the element normal, k is wavenumber and λ is wavelength.

However, note that in the expression above $\operatorname{sinc} x = \frac{\sin x}{x}$, but in both Matlab and Python the built in `sinc` function is defined as $\operatorname{sinc} x = \frac{\sin \pi x}{\pi x}$, which means you need to divide the argument by π to get the expected behaviour.

Finally, if you are using a student version of Matlab, it may not have the toolbox containing the `sinc` function installed, so a version with identical functionality is provided by the file `fn_sinc`.