2017 Ultrasonic transduction course

Lab: model of a monochromatic ultrasonic beam

By Jerzy Dziewierz,

Centre for Ultrasonic Engineering, University of Strathclyde, 2017

## Introduction: Why bother?

The job of a scientist is to find ways to predict the future. The job of an engineer is to predict the future change the future in his favour. You are scientist-engineers, meaning that You have a special mission: You are the people that are especially good in changing the future by virtue of understanding what only a few people understand.

To predict the future, we can describe the world we live in using a theory. Then, we can evaluate the consequences of this theory, either in our heads or use some tools external to us.

Now, there is some good news and some bad news.

The good news is that by now, we have a pretty darn good model of the world we live in – at least up to a scale of the earth. It’s called quantum mechanics. That’s right. You plug the numbers into the equations, run the equations, and it turns out that the experimental data will match the predicted results very well. It’s safe to say that quantum mechanics is very accurate.

The bad news is that these equations are complicated. Difficult.

To say that something is difficult is simply to say that it will cost a lot to do it.

For an engineer, it is easy to make things expensive. The genius is in making things cheap.

Here lies an opportunity: if you have a cheap model, one that is just good enough, you can predict what will happen just accurately enough, then your product will be just good enough for cheap.

It turns out that there is a vast market for making simplified models of reality. They enable us to predict the future for cheap.

Today, I am going to give You one of such models – the model of a monochromatic acoustic beam. It is based on a theory by Mr Huygens (1678) and Mr Fresnel (1816). You can look up what they found on Wikipedia.

https://en.wikipedia.org/wiki/Huygens%E2%80%93Fresnel\_principle

Now, before we begin, please have a clear understanding of one common fallacy that I have seen multiple times:

It is easy to observe that more accurate models typically cost more to evaluate. The error lies in the fact that theory of reality only works in one direction: but if you only know that the price of running a model is high, it does not mean that you will get more value out of it.

Ok, let’s get to work now.

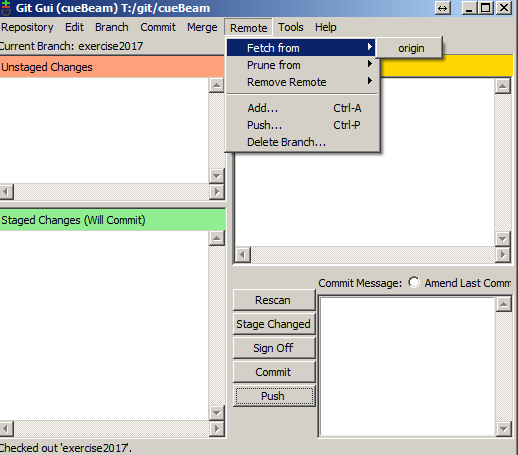
# Load the latest update from github, using git:

The model that I am developing is hosted for Your pleasure on a public repository on GitHub.

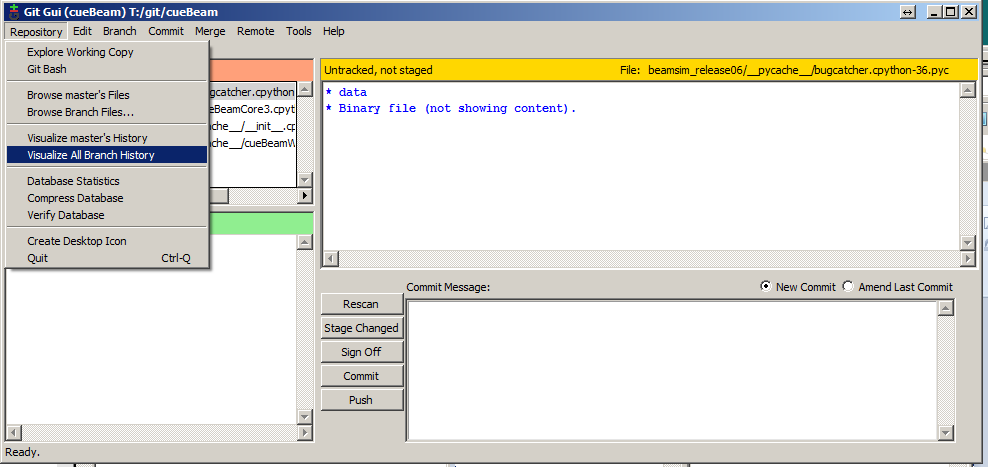
If you don't know about git, well, this is not a course about git, but I can recommend to You that You learn.

For now, you can download the latest version of the code by using this commands:

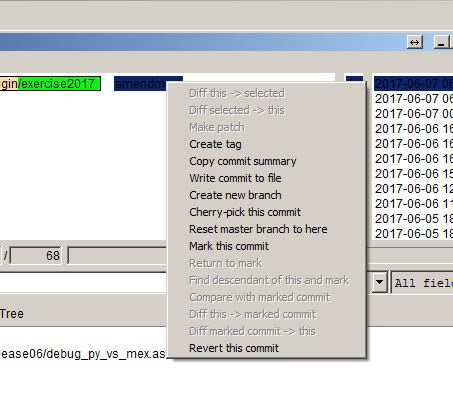
* From the desktop, open “Git GUI”
* Open recent “cueBeam”
* Command “Remote->Fetch From->Origin”



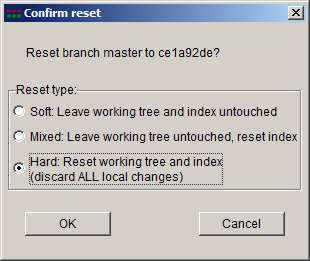
* Command “Branch->Checkout…”
* Select “Master”, then OK “Checkout”.
* Command “Repository->Visualise All Branch History”



* In the left-top subwindow select the top line, click the part of the line text that has no green box. You might need to resize your window or subwindow to get there.
* Right click on that text: “right click -> Reset master branch to here”



* Select “Hard: Reset…. Discard all local changes” And OK



* Close the gitk window.
* In the main GIT GUI window press “rescan”. The subwindows should be empty. If they aren’t, something went wrong.

**Now, open Matlab.**

In Matlab, change the current folder to the “c:/cueBeam/beamsim\_relase06/”, or the folder where your actual files are.

Run “cueBeam\_linear\_array\_basic”

If we are lucky, you will see some figures pop up.

If there are errors, this is the moment of truth for me.

I Owe You an apology at this time – the software is not complete, and it is not as pretty inside as I would like. The reason is that it used to be written in vanilla Matlab, and I am now moving towards Python. However, to sweeten the deal, you get it for free.

# Explore the model parameters

The model inputs are organised into 4 clusters:

**Environment** –

The package assumes that the propagation medium is uniform and homogenous. The only parameter that you set there is wave\_velocity. Remaining parameters are calculated by the package.

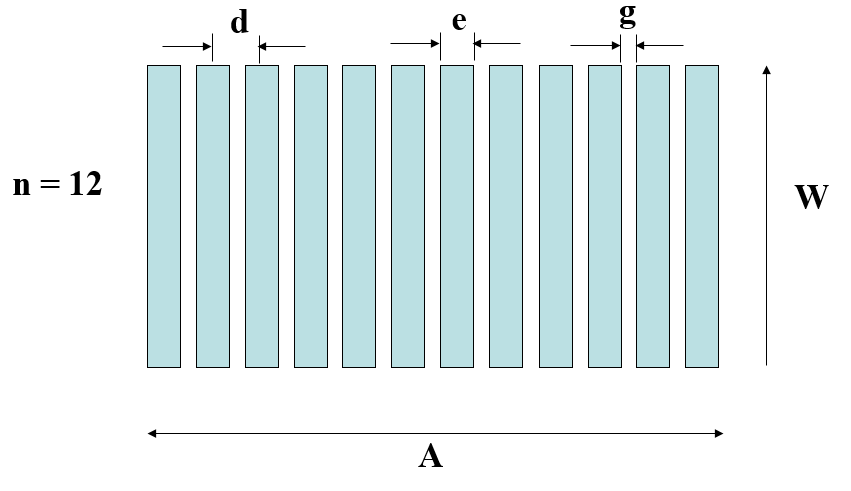
**Beam** –

The default package assumes that you want to use all the probe elements to do focused azimuthal steering (focussed azimuthal B-scan). Specify alpha\_rotation (steering of the beam) and focal\_distance.

Secondly, the display\_limit\_db controls the axes range in the output visualisation figures; otherwise, it has no effect on the core of the simulation.

**Probe** –

The default is to assume that you want a linear phased array probe. Although the package can simulate other probe types, the “process\_linear\_array” script only deals with a linear probe and related properties. The parameters you set are best explained by the drawing:



**Simulation** -

Calculation options:

* Lambert\_map\_density, xy\_resolution: the resolution of the pressure field images.
* Xy\_z\_extent, xy\_z0, xy\_x\_extent : these parameters control the XY section image size
* Verbose, doplots, doprints, papersize – select whether to display and save the outputs. Note that the outputs can be automatically saved as .png images to a folder specified by “prefix” – for inclusion in Your technical report.
* Do3DPlot1: create a ‘preview quality’ 3D drawing with two cross-sections of the space: the XY cross-section and the Lambert (hemisphere) cross-section. This is relatively quick to do, but you might want to disable it to make it even faster (e.g. for purposes of parameter sweep or optimisation).
* Do3DBeam: this will enable a full volumetric calculation. Warning! This is a comparatively time-consuming operation – we DO NOT DO THIS during the labs due to limited computation power available. You can try this on your own GPU-enabled computer. Ask me for more details.
* doXZBeamSection, do LambertSection – enabled by default, you might want to disable them to make a parameter sweep faster (if you do not need that particular section to complete your parameter sweep)
* XZBeamSection\_Z, lambert\_radius – where from in space to take the line-section. This is used to calculate the beam width and side lobe level. By default, the line section is taken at the focal distance. Note that for the Lambert section, the section is controlled by the Lambert hemisphere radius.
* Prefix - the file name prefix for the saved results. Choose wisely, as old files are being overwritten with no warning.
* LineSmoothing – On some systems, this will enable line anti-aliasing. You might want to try that on final simulation run, as it will slow down the saving process by quite a bit.

Then, “cueBeam.process\_linear\_array” launches the simulation script.

The simulation script does following actions:

* Prepares Matlab for display and saving the results
* Calculates calculable properties like wavenumber and internal calculation resolution, focal point XYZ coordinates,
* Calculates some basic statistics about the probe, like the aperture, near field size e.t.c.
* Generates a detailed description of the probe - terms of radiation points. See the Huygen’s theorem for more explanation. At this point, apodization is applied (if any), and delay laws are calculated.
* Does basic visualisation of the delay laws
* Converts the probe description into the core description format that is suitable for accelerated parallel calculation on the GPU. Then calls the GPU code to do the calculations. See the source code for more details. There is a lambert-map section and the XY-section.
* After the pressure field is calculated by the GPU, it generates visualisation plots
* Calculates statistics about the beam:
  + beamwidth\_XZ: width of the beam in meters in the XZ section
  + IntegratedSidelobe\_XLine: -23.0647 : average side lobe amplitude
  + peakSidelobe\_XLine: -12.9917 : peak side lobe amplitude
  + beamwidth\_lambert: 6.6313 : width of the beam in degrees (180deg = full hemisphere) – taken from the lambert hemisphere
  + IntegratedSidelobe\_lambert: -22.1738. As previously, mean and peak side lobe.
  + peakSidelobe\_lambert: -9.4488

Note that for most purposes, XY section is easier to understand and explain, but the Lambert section is more accurate and realistic measurement. This is because the points on the hemisphere are (approximately) at an equal time-distance from the probe, meaning that the time domain signals coming from that hemisphere are not separable (overlaid). It is therefore particularly important to get a good (low) side lobe level in the Lambert hemispherical section.

The result of the model is saved into variable “result”. Explore the variables available there.

# Take a look at how to make a parameter sweep

Examine the files:

cueBeam\_linear\_array\_pitch\_sweep

cueBeam\_linear\_array\_angle\_sweep

These demonstrate how would one approach executing a parameter sweep.

One can create an optimiser harness in a similar way.

Note! When building the optimiser harness, take care to specify a sensible minimal step size for each controllable parameter.

# Execute the design process

When you feel that you are familiar with the meaning of the input and output parameters, it is time to do the design according to specification.

**For each student group, choose at random:**

* Desired side lobe level – randomly between -30dB to -10dB
* Desired steering range – randomly between 30deg sweep to 70deg sweep.
  + Note, for single stepping and visualisation, steer the beam to the extreme.
* Focal distance – randomly between 10mm to 40mm

Make a note of your choice – you have to specify it in the report.

**For given design parameters, optimise the following goal:**

* Maximise resolution == minimize the beam width.

**For given design parameters, observe the following constraints:**

* Number of elements – 16
* Frequency of operation: 1MHz
* Medium wave velocity – 5600m/s
* Peak side lobe lower or at most as specified (as in the ‘choose at random’ section).
* No grating lobe within the steering range

**Obtain the result by controlling:**

* The element pitch “d”
* The element width “e”
* The passive aperture “W”
* Apodization type and coefficients

**You are free to do either of the following:**

* Random walk – change inputs and evaluate the outputs by hand
* Parameter sweep. Write a test harness and result visualizer. Please use a reasonable amount of calls to the solver; for example, the element width step size can be 0.2mm, but not 0.01mm
* Automated optimiser – use method of your choice. I’d recommend a simplex-descend algorithm. This is entirely optional. Again, please be careful to specify the minimum step size for each of the variables.

**Hints:**

* It might not be worth to explore \*all\* possible options and parameter combinations. Use Your Engineering judgement to decide on what to do best. Remember, the genius lies in making things cheap.
* For your presentation:
  + make it very clear what input constraints you were designing for;
  + what final result has been achieved
  + bonus question: what is the dominant controlling parameter (in other words, controlling which parameter affects the result the most?)