Tallulah GILLIARD

12/06 – 31/08

2017

INTERNSHIP REPORT

IUT de Bordeaux

Département Informatique, Année Spéciale

Finding Patterns in the acoustic levitation system with deep-learning techniques

Sriram Subramanian, Gianluca Memoli

INTERACT Lab

Sussex University

Brighton

IUT Département Informatique Bordeaux

interact LAB

Finding Patterns in the acoustic levitation system with deep-learning techniques

Supervisors : Sriram Subramanian, Gianluca Memoli
Student : Tallulah Gilliard

Abstract

Haptics are an important topic to focus on in the human computer interfaces and surfaces field. Haptics are use in the ultrahaptics board, that can be layered by “metamaterial bricks” which modify the wavelength proprieties that is produced by the board.
The goal is to improve the model developed in another paper which study how a wavelength can be delayed by going through some “maze” and adapt it to our case of metamaterials bricks made for the ultrahaptics board.
The way reaches that goal is done by using computer science and mathematical skills combined with languages and softwares like Python, Java and COMSOL.
Here we want to obtain the phase delay of a wavelength going through bricks, then adapt it to other bricks to be able to use deep-learning techniques to find new better bricks. We explore several way to get to the expected results to finally get the right phase delay with COMSOL at the end of this internship.

Acknowledgements

This internship has been a really good experience to me. I learned a lot and improve myself on different aspects of computer science and mathematics skills. I ran through several different articles, coding languages and knowledges. All of theses made that first internship a rich and positive experience.
That’s why I would like to thank in the first place my internship supervisor, Sriram Subramanian for having gave me the opportunity to perform this internship.

I would like to warmly thank Gianluca Memoli who have been my second supervisor on that whole project, who helped me and challenged me wisely at each step of my internship.
The whole team of INTERACT Lab has also been there for me from my first day by warmly welcoming me to my last day by organizing a farewell for my departure. There was always someone to help me when I was a bit lost. Also, I felt really well integrated by being invited to join the journal club of the SCHI lab or any social after work activity.

Thanks as well to my French supervisor Sophie Cartier who helps me to improve my English during the year and helped me with my email exchanges before the beginning of my internship.
None of these steps would have been possible without the support of the IUT pedagogic team who has helped me following my projects (dreams) from the moment when I step in the Année Spéciale class. Indeed, that degree was exactly the computer science boost that I needed after my bachelor degree in Maths Applied to Cognitive Science.
It’s also thanks to both of that technological degree and internship that I’m accepted in the Human Computer Interface and Design master degree that I’ll proudly start in September !

Another person that I would like to thank for having played an important role, is Camille Jeunet, researcher in Brain-Computer Interfaces, who have talked me about the Ultrahaptics company 3 years ago already and for have introduced me to Sriram during her (brilliant!) thesis defense in December 2016 at Inria Potioc team.

I’m being grateful to all the people that I met in Brighton who host me, proposed me activities or simply spend time with me, this has played an important part in my comfort and English improvement.
Last but not least, thank to my friends and family who have supported me from aboard by sending me messages and letters during my whole stay.

Table of contents

Introduction

In the human computer interfaces field, haptics are an important topic to focus on. It appears in everyday life of smartphones, tablets or tactile interfaces device users. Most of those devices uses the physical proprieties of touch, but by adding the notion of kinesthesia (movement) and proprioception (location and orientation), you'll be able to provide a more complete haptic feedback and have a more genuine experience.

I've discovered interactives surfaces and human computer interaction dedicated to haptic devices during my bachelor degree in Maths applied to Cognitive Science, until then they have kept my strongest interest. I've attended to a thesis defense at Inria Bordeaux and had the opportunity to meet Sriram Subramanian, co-founder of Ultrahaptics.

This internship of technical degree in computer science (DUT informatique année spéciale) was from the 12th of june to 31th of august 2017. The goal of my work was to work on the "metamaterial bricks", reproduce results from a paper and find some new results and then use some deep-learning techniques to find new "bricks" adapted to our device.

Through this report, I will first describe my working environment inside the INTERACT Lab, then how I've been involved in the project with a knowledge appropriation followed by the core of the project which was coding with python ad the beginning a bit of using specific excel fuctions, then learning to use a new software named COMSOL and learn to use it with java.

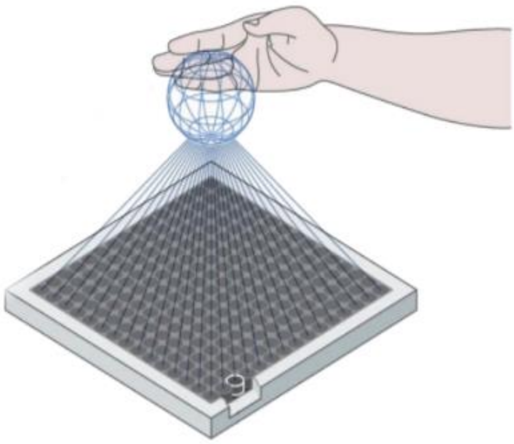
Part 1 : The Ultrahaptics company and the INTERACT Lab

The places & peoples

Ultrahaptics, is an haptic device company founded in 2013 and based in Bristol that produces technology that uses ultrasound waves technology to construct 3D objects in the air that users can feel.

Hand gestures are tracked by a Leap Motion controller (hand sensor tracker). The transducers information is processed by a driver card connected to a Computer system. An application based on the algorithm developed by the University of Bristol permit the synchronization of images with the device.

The field of applications is quite wide, through aeronautics, automotive industry, medical industry, video games or education.



The Interact Lab of the University of Sussex is a research lab within the School of Engineering and Informatics. The lab is part of the creative technology group and brings together staff and post-graduate researchers from Informatics, physics, engineering and design to pursue research in human-computer interaction. It is related to Ultrahaptics because of researchers' relationships (Sriram Subramanian Professor of Informatics at Sussex University is co-founder of Ultrahaptics)

People in the INTERACT Lab use Ultrahaptics devices but do not work for the company. They do research about various topics related to haptics, wavelength, mechanics, physics and technology.

People of the INTERACT Lab

Sriram SUBRAMANIAN

Sriram is Professor of Informatics at Sussex University and Co-founder of Ultrahaptics. He brings new projects and ideas to the team, manages who work on which project depending on everyone skills, he negotiates deals with other companies, he also travels a lot around the world to attend or give conferences, present papers or projects to promote the visibility of the INTERACT Lab, Sussex University and Ultrahaptics.

Gianluca MEMOLI

Gianluca is a newly appointed lecturer specialized in acoustics. His primary interest is to use acoustic metamaterials to facilitate multi-sensory human-computer interactions involving touch and sound. He is also interested in sound perception.

Diego MARTINEZ PLASENCIA

Diego is a lecturer working on novel formats for 3D displays. His research interests focus on systems allowing 3D visualization, interaction and collaboration without involving additional instrumentation of the user (i.e. no VR headsets, 3D glasses, etc.), usually involving various types of autostereoscopic displays, fog displays or even soap bubbles.

Spyros POLYCHRONOPOULOS

Spyros is a researcher working on “Acoustic Levitation and Computer Human Interaction” research project, creating DSP/FEM models, real world experiments, novelty in approach, etc. His background is physics and electrical engineering and worked as an acoustic consultant for 3 years. He is also a music composer and has released 15 albums.

William FRIER

William is a PhD student. His research work focuses on the field of Human-Computer Interaction and mid-air tactile display. He’s interested in exploring how to reproduce tangible interaction in mid-air using existing mid-air tactile display. One example is how to reproduce the feeling of a physical button in mid-air (shape, roughness and stiffness) using recent breakthrough in psychophysics or biomechanics about touch perception.

Yutaka TOKUDA

Yutaka is a Research Fellow of Shape Changing 3D Displays. He has worked on holographic mid-air displays, shape changing 3D fog displays, mobile 3D telepresence robot and many digital public art projects. He is currently trying to make liquid metal robots like T-1000 in the famous movie Terminator 2.

Louis JACKOWSKI

Louis is an Embedded Software Engineer. He is responsible for writing software related to acoustic research, particularly metamaterials, which are used for haptic feedback and directional sound.

Adili NORASIKIN

Adili is a PhD Student, he is investigating about mid-air displays, and he focuses on physical matter displays. His studies involve different approaches such fog in airflow field, and levitated matters inside ultrasound field. One example of his project titled “Mistform: Adaptive Shape Changing Fog Screens” presented in Computer CHI2017 conference in Denver, Colorado.

Patricia CORNELIO

Patricia is a PhD student. Her research work is focused in the field of Human-Computer Interaction and the Sense of Agency. She’s interested in exploring how the experience of agency can be influenced by HCI interaction paradigms such as touch-less systems, mid-air interfaces and Virtual Reality (VR), and how this experience can be implicitly measured in novel application scenarios.

Luis BERNA

Luis is the research technician in the lab. He returns to education after 10 years working in industry as a mechanical/electrical eng. Active member of various local maker communities, he has always had a passion for electronics which now uses to help the lab members to create prototypes and provide advice of current projects. Interested on too many fields to list them and soon starting a PhD on "Novel formats for 3D User Interfaces".

Roberto MONTANO

Roberto is a PhD Student whose research is focused in immersive VR, more specifically to increase the feeling of subjective factors such as Ergonomic comfort, natural interaction and navigation by using methods like ergonomic/spatial re-targeting, drift correction and optimization algorithms.

Stephen BECKETT

Stephen is the lab’s Artist-in-Residence and Research Assistant. He works with researchers in the lab to create installations, experiences and events that reimagine the lab’s unique technology in new settings and for new audiences.

Luis VELOSO

Luis is a photographer / videographer who helps visually improve how to explain the researcher's projects at Interact Lab with illustrations, photographs and videos destined for the scientific community. Also, he designing the corporative image of the stand exhibit in different scientist fairs where Interact lab has participated.

Lucy ARNOLD

Lucy is Project Administrator, she organizes travel and logistics for conferences and events, manage deliveries, liaise with Finance over purchasing and payments, co-ordinate with other University departments and external guests/speakers and provide administrative support to Professor Subramanian and the rest of the Interact Lab team.

Around and inside the INTERACT Lab

Chichester Building :
Offices & Sriram Office

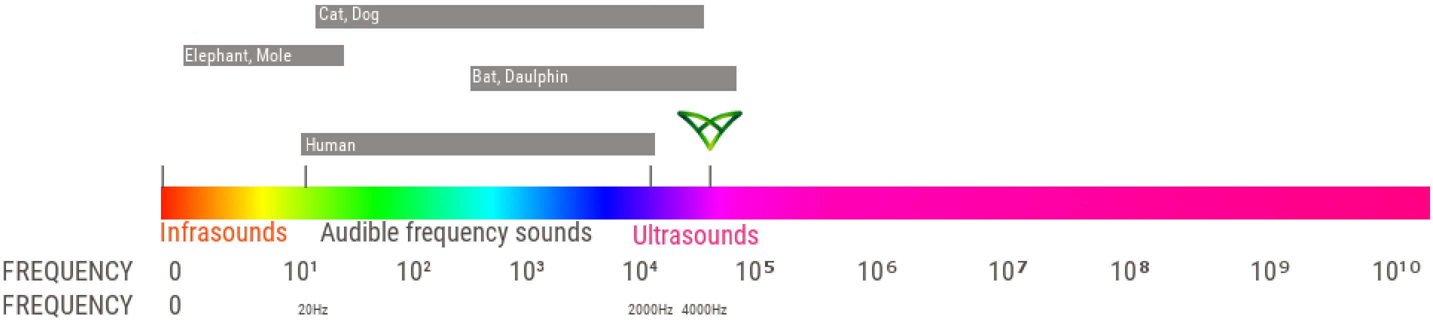
Shawcross Building :
School of Engineering and informatics. IT Services

The device

The Transducer array send ultrasound at the frequency of 40kHtz, a frequency which is not audible by human ear. It’s composed by 16*16 transducers (≈ speakers) (MA40S4S, Murata Electronics, Japan)

ⓐ *The Transducer Array (or Ultrahaptic board)*

Representation of a “virtual” sphere

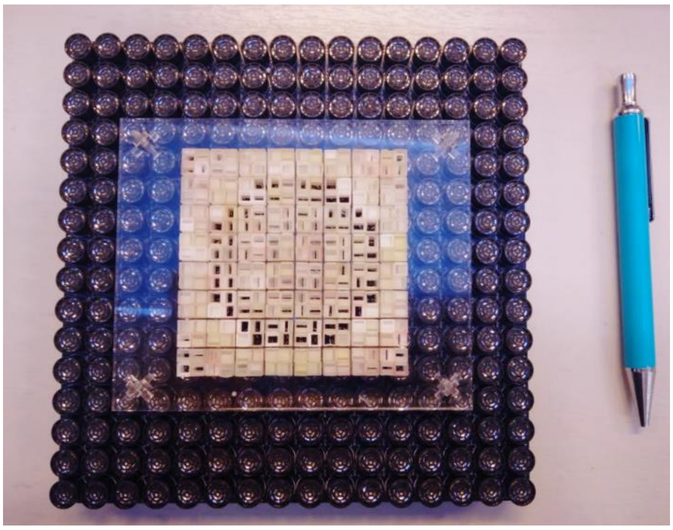


Graph bar of sounds and their frequency

The transducer sends those ultrasounds to precise focal points in order to create “virtual” shapes, objects or focal points that you can feel or use to levitate small objects like small polystyrene bead.

At the moment, there is two ways to create theses haptic stimuli :

- ⓐ With the transducer array by itself (details just below) which allows a dynamic interactive creation of 3D virtual “air” shapes
- ⓑ With the metamaterials bricks (details in the following part “METHODS Part 1 / Metamaterial bricks”) allowing a reduced surface (using a 8*8 array instead a 16*16 for the same results) but that create static 3D virtual “air” shapes

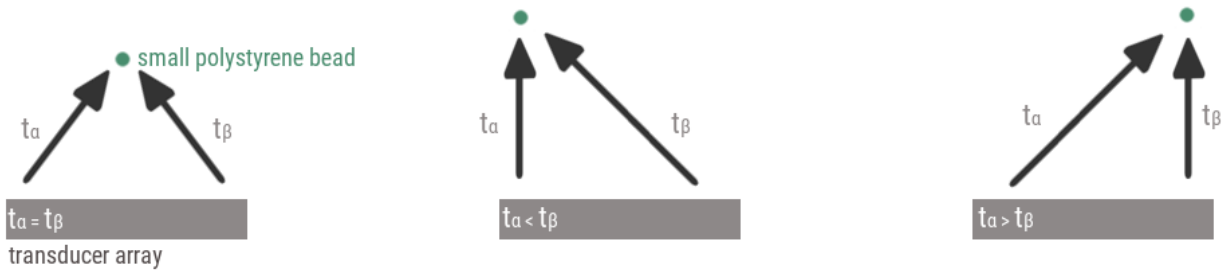


ⓑ *The Transducer Array with the metamaterial layer*

Transducer array to levitation

The whole concept is based on wavelength delay to create focal points and being able to levitate small objects (objects about 5 milligrams). The speakers emit the sound at different moments, this permit to create different “focal points” (point where the waves intersect).

For example, in the first image on the left, just below, the speakers emit a wave at a same time so they intersect in the center of the array.

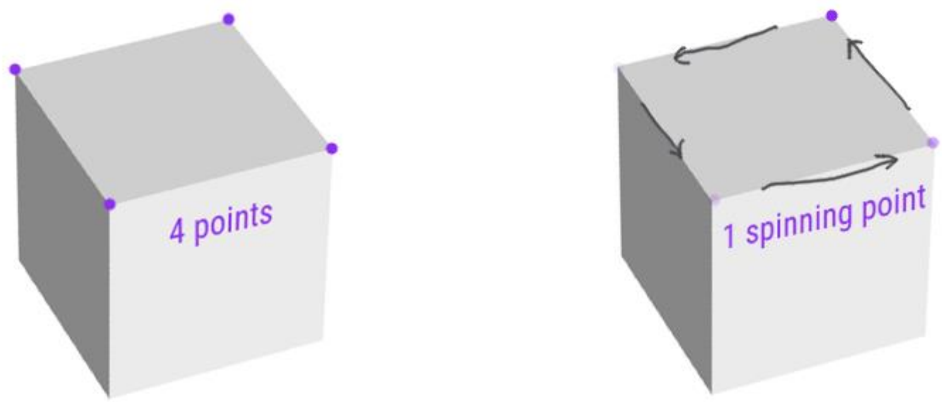


3 different cases of wavelength delay situations (and bead position related)

Transducer array to create shapes

The leap motion captor permit to give a feedback of the hand position above the array and interact with it like you would do with a classic shape (like a cube, a ball or a pyramid).

There’s two ways to create this. By creating a certain amount of points or by creating one point that moves according a certain shape



The two different techniques

This technology has a large field of applications in games (VR, controllers), aeronautics or automobile technologies (dashboard design, haptics controlers), industry (collaborative tables, architecture), or best devices for customers (“intelligent” kitchen, dj-ing, smart objects, application on interactive screens). This is still a quite new innovative field, that’s why there is a lot of applications and related technologies to develop.
The INTERACT Lab work on a wide range of projects linked to physics, electronics, mechanics, like the metamaterials bricks applied to the ultrahaptics board on which my internship was focused on. In the following part of my report, I’ll develop this aspect.

Part 2 : My involvement

OBJECTIVES

The Goal of my work was to improve the model developed in 2014 by Yong Li et al. [1] which study how a wavelength can be delayed by going through some "maze" and adapt it to our case of metamaterials bricks made for the ultrahaptics board. The advantage of adapting that article to our case was to not rely fully on Finite Element Methods (FEM) which is a calculation technique for simulating the field of the chosen wavelength. This FEM method take a lot of computing time, the goal was to reduce that computing time by reducing the possible solutions by using a deep learning approach

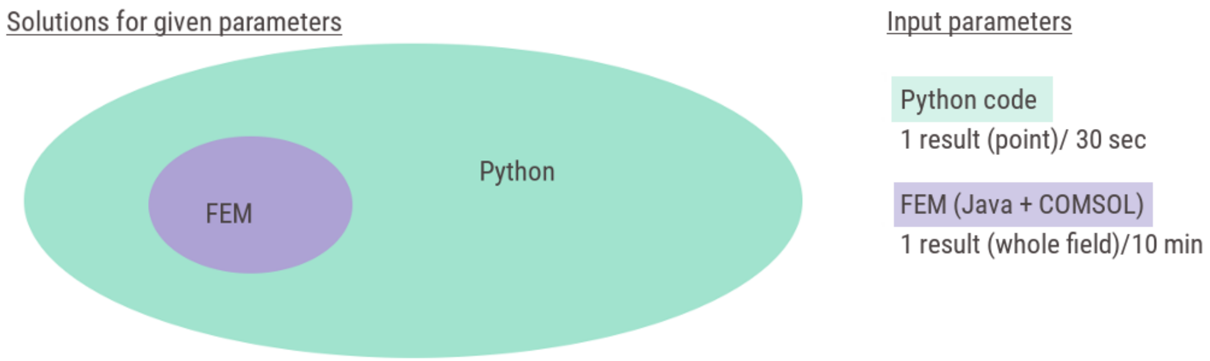
We wanted to find the metamaterial bricks with the best transmission wave (less loss as possible) by making some variations of the brick parameters. In the article [2] only two parameters are variables : the bar length and the inter-bar spacing.

The parameters that we want to modify are :

- Number of bars (m,n)
- Frequency
- Bar parameters (w,d,t, l)

Check transmission and phase depending on theses parameters.

Due to the amount of parameters, there's an important number of combinations.
To judge if the transmission is correct, making some simulations are required. However, a COMSOL (finite element analysis (FEM) solver and simulation software for various physics and engineering applications) takes about 30 min to generate the whole field and this would take 30 min multiplied by the number of combinations and would be definitely too long.
That's why the goal is to have a first analysis on the field of all solutions, browse the more relevant ones and do the COMSOL simulations on them.



At the end of the process, we would like to 3D print the bricks obtained by simulations and test them on the device.

INTERNSHIP OBJECTIVES TABLE

(= done, = ongoing, = to do)

Tasks		Deadline	Progress
Read the articles, understand processes, state of the art		19/06	
Reproduce the results obtained in the article [1]	Try to improve Louis's code to obtain the article results	22/06	
	(or) Start from scratch to test the Simpson integration with by implementing a simple function ($ax + b$)	23/06	
	Increase the level of complexity to reach $\phi_n(y)$ (page 3 (11))	27/06	
	Implement the rest of the page 4 to reach $\tilde{\beta} = \rho_0 c_0 / Z_s = i\tan(\phi/2)$ (transmission at the exit) and check if we obtain the same results	7/07	
Creating a general model (mostly by differencing d_x and d_y which are a same sized d in [1])		19/07	
Test this model without the boundary conditions (way back) but with the transmission like in the article [2]		26/07	
Apply this general model of transmission to the article [2] to obtain the same results		4/08	
Finding finite element analysis (FEM)	COMSOL : learn basis (create a brick, a mesh, a simulation)	?	
	reproduce results from paper [1] and use COMSOL with Java by executing .java files	?	
	Generalize COMSOL bricks with parameters, generate some automatically by changing parameters	?	
	…	?	
	Deep learning ? find bricks	?	
…		?	
3D print and test the new(s) brick(s) we've found with DL/COMSOL		30/08	

[1] Yong Li, Xue Jiang, Rui-qi Li, Bin Liang, Xin-ye Zou, Lei-lei Yin, and Jian-chun Cheng1, *Experimental Realization of Full Control of Reflected Waves with Subwavelength Acoustic Metasurfaces*, PHYSICAL REVIEW APPLIED **2**, 064002 (2014)

[2] Gianluca Memoli , Mihai Caleap, Michihiro Asakawa , Deepak R. Sahoo , Bruce W. Drinkwater & Sriram Subramanian, *Metamaterial bricks and quantization of meta-surfaces*, Nature Communications **8**, 14608, (2017)

METHODS – part A

How to improve this technology : State of art & article synthesis

To fully understand the work to do, some preliminary researches have been necessary. I've read some papers related to wavelength, acoustics and metamaterials. The essentials ones are below and also in "references" :

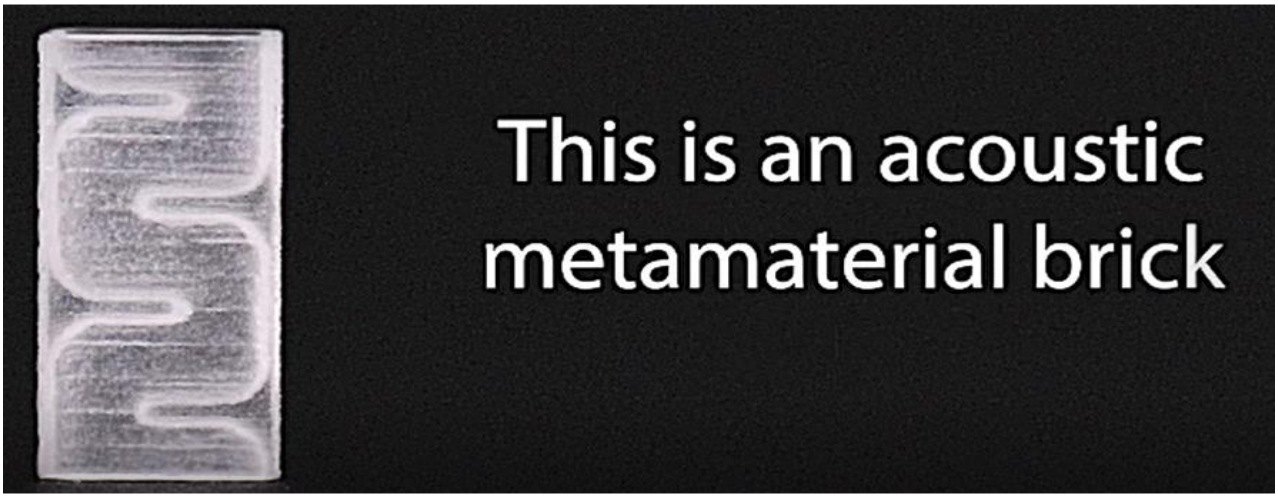
[1] Yong Li, Xue Jiang, Rui-qi Li, Bin Liang, Xin-ye Zou, Lei-lei Yin, and Jian-chun Cheng1, Experimental Realization of Full Control of Reflected Waves with Subwavelength Acoustic Metasurfaces, PHYSICAL REVIEW APPLIED 2, 064002 (2014)

[2] Gianluca Memoli , Mihai Caleap, Michihiro Asakawa , Deepak R. Sahoo , Bruce W. Drinkwater & Sriram Subramanian, Metamaterial bricks and quantization of meta-surfaces, Nature Communications 8, 14608, (2017)

With those basis, here is some summary of what are metamaterials bricks and how they work :

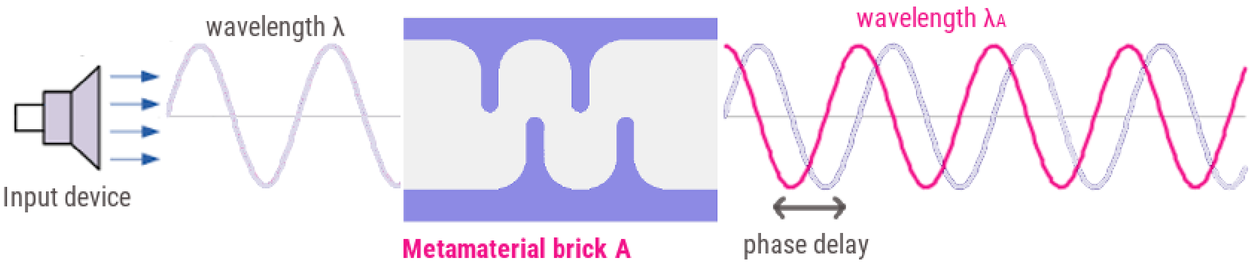
Metamaterial bricks

Recent studies [1] have shown that we can use "maze-bricks" called "metamaterials brick" to modify the wavelength. In simple words and apply to our case, the "air" (ultrasound) takes longer to get out through a maze than getting out of a simple pipe. It gives a delayed wavelength at the exit, so the physic proprieties are modified.
In Wikipedia words "*A metamaterial (from the Greek word μετά meta, meaning "beyond") is a material engineered to have a property that is not found in nature. They are made from assemblies of multiple elements fashioned from composite materials such as metals or plastics. The materials are usually arranged in repeating patterns, at scales that are smaller than the wavelengths of the phenomena they influence. Metamaterials derive their properties not from the properties of the base materials, but from their newly designed structures. Their precise shape, geometry, size, orientation and arrangement gives them their smart properties capable of manipulating waves : by blocking, absorbing, enhancing, or bending waves, to achieve benefits that go beyond what is possible with conventional materials.*"



What's a metamaterial brick and how does it modify the wavelength ?

The metamaterial bricks were manufactured from thermoplastics using a 3D printer (ProJet HD 3000 Plus), which has a print resolution of 25 µm.
Those bricks have the propriety to modify the wavelength. The wavelength enter in the brick with a certain phase φ, go through it, and depending on the shape of it, it affect wavelength phase because of viscosity and how complex is the maze. The wavelength goes out with a phase delay φ' modulo 2π*

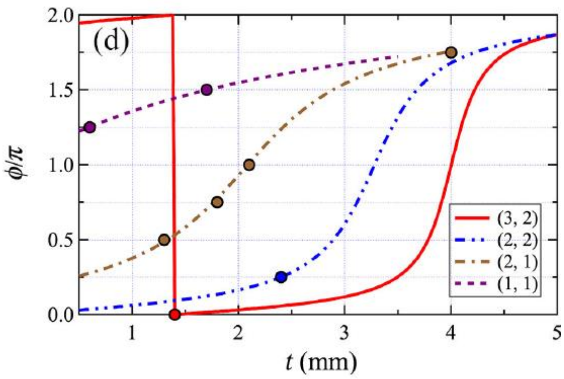


Phase delay of a wavelength going through a metamaterial brick

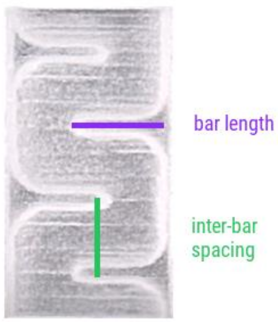
* Based on the fact that different wavelength phases appears depending on the shape of the brick, there's theoretically as different type of wavelength as different types of bricks. But when a wavelength has a phase of 2π it's back to where it started. So, we can obviously say that the phase range shifting between 0 and 2π.

How choosing the bricks ?

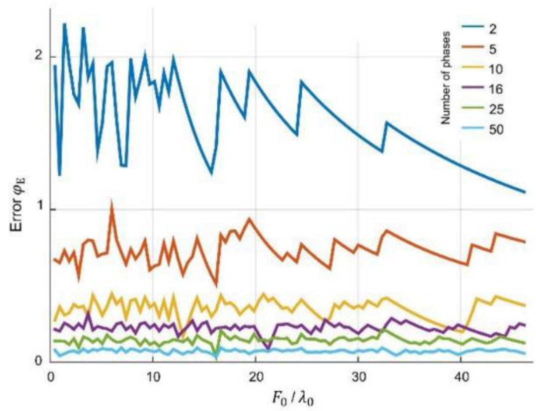
In the Physical Review Applied paper [1] they took 8 bricks with a phase shift depending of a Pi value (from 0 to 360 with a step of 45).
Graph of chosen bricks for the article [1] →



In the article [2], 1200 simulations (30*40) have been made by variating the bar length of the brick b_l or l (30 points) and the inter-bar spacing b_s or d (40 points).
↓ Image of the brick and variating parameters just below.

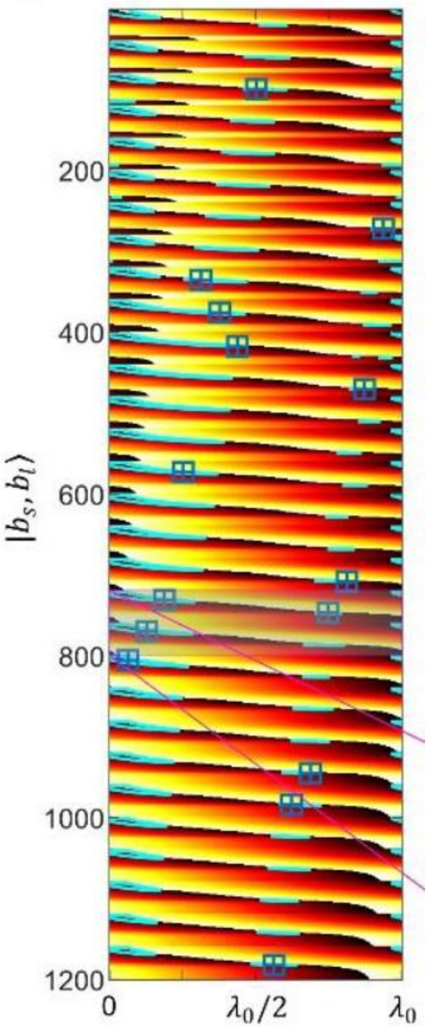


The 16 bricks with the "best" transmission have been chose (with less than 5% of loss of the initial wave).
The decision to take 16 of them instead of 8 (like in the article [1]) or another number is due to the error rank (curves graph on the right, purple one →) which stabilize at 16.



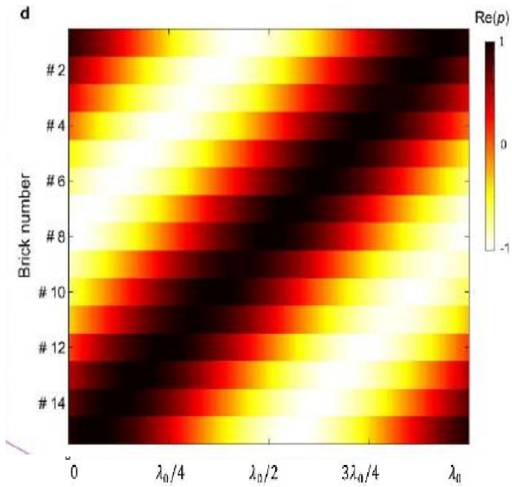
Those 16 bricks (#) are also depending of the values of Pi (φ) in degrees, permitting to cover all the values.

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
φ	**0**	22,5	**45**	67,5	**90**	112,5	**135**	157,5	**180**	202,5	**225**	247,5	**270**	292,5	**315**	337,5

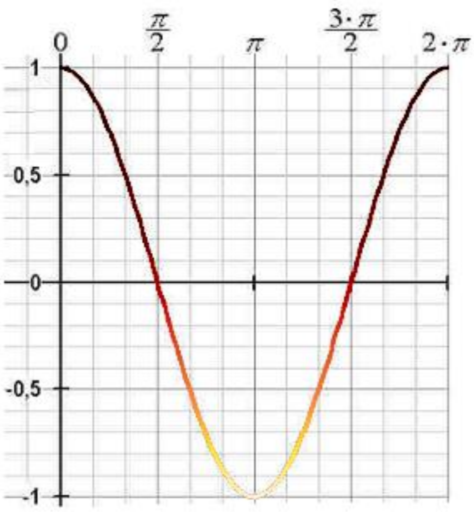


← Graphic of the 1200 simulations.
In clear blue are the best simulations (with less than 5% of loss of the initial wave).
The dark blue squares are the 16 chosen bricks.

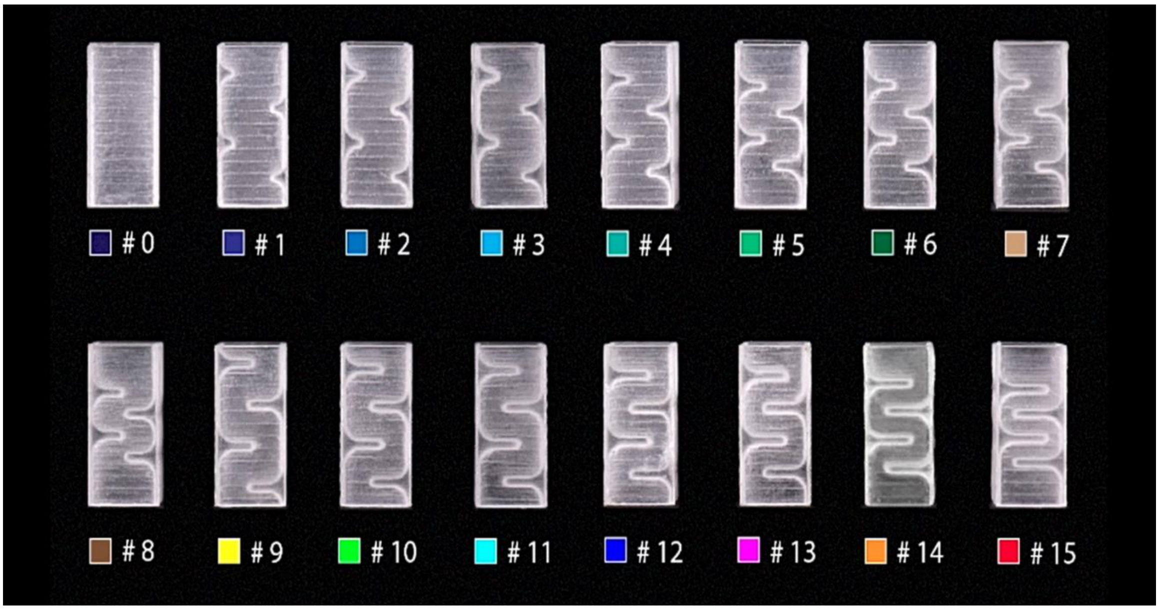
Graphic of the 16 bricks and their associated wavelength →
In this graphic, it's easy to see the delay. The brick #0 has almost the same output wavelength than the brick #15 (due to periodicity)



↓ Corresponding colors to Pi values on the graphs



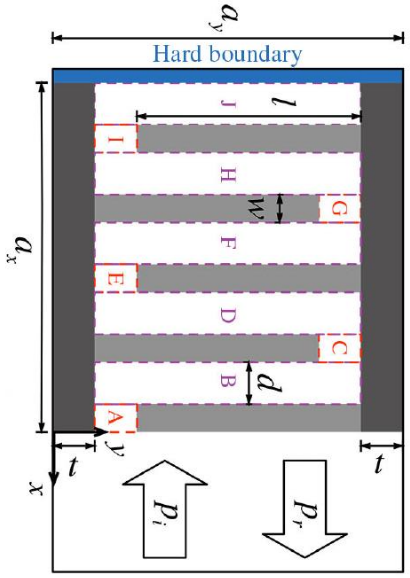
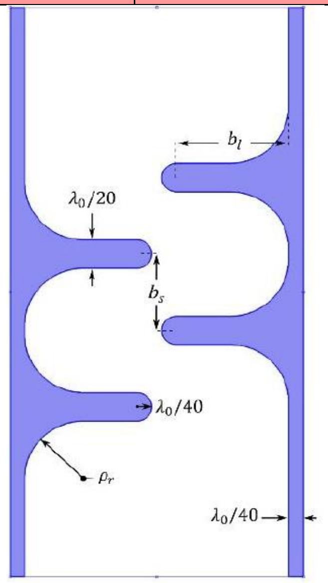
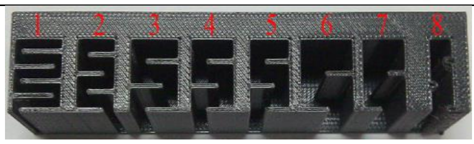
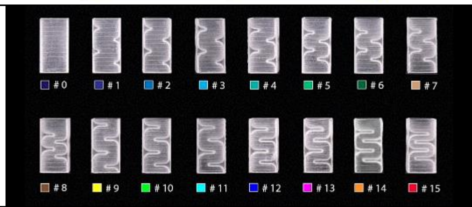
The 16 type of bricks :



Based on that, there's 16 types of bricks that have been created (image above) the changing parameters are detailed in the table below.

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15
Target phase angles														
22.5°	45°	67.5°	90°	112.5°	135°	157.5°	180°	202.5°	225°	247.5°	270°	292.5	315°	337.5°
Actual phase angles														
23.3°	47.5°	67.4°	89.7°	115.6°	134.4°	159.3°	177.3°	204.8°	226.2°	246.4°	271.4°	295.3°	315°	335.3°
Transmission coefficient (magnitude)														
0.987	0.999	0.973	1.0	0.999	0.993	0.999	0.997	0.963	1.0	0.977	1.0	0.999	1.0	1.0
Bar length, b_ℓ / λ_0														
0.062	0.092	0.112	0.132	0.152	0.162	0.171	0.191	0.221	0.241	0.251	0.271	0.281	0.301	0.321
Inter-bar spacing, b_s / λ_0														
0.216	0.212	0.207	0.189	0.161	0.166	0.171	0.134	0.257	0.234	0.230	0.207	0.203	0.175	0.152
Fillet radius, ρ_r / λ_0														
0.062	0.092	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Supplementary Table 1 | Key characteristics of the metamaterial bricks used in this study.

PARAMETERS (values in mm)		[1]		[2]	
Frequency (Htz)	f	3430		40000	
Wavelength (in meters)	λ_0	0,1		343/ f	0,00858
Brick height	X	$\lambda_0/8$	12,50	λ_0	0,00858
Brick width	Y	$\lambda_0/8$	12,50	$\lambda_0/2$	0,0042875
Brick depth	Z	/	/	$\lambda_0/2$	0,0042875
Outer width	t	t		$\lambda_0/40$	0,000214375
Bar length	l	$l = y - 2t - d$		b_l	
Bar width	w		1	$\lambda_0/20$	0,00042875
Inter-bar spacing	d	$d = \frac{x}{m+n} - w$		b_s	
Bar roundness		/	/	$\lambda_0/40$	0,000214375
Fillet radius		/	/	ρ_r	
Number of bars on left boundary	m			m	2
Number of bars on right boundary	n			n	2
Bricks figure					
Bricks image					

- fixed parameters
- fixed depending parameters
- variables
- depending variables
- results

METHODS – part B

How to improve this technology : Implementation and coding

Initial code

Python

First, I had to understand, rewrite and complete an existing code written by Louis the Embedded Software Engineer. This code was implementing only the first part of the maths from the paper about metasurfaces [1]. It was supposed to give the wavelength delay once passed through the brick.
The aim of it was to reproduce numerically what was describe analytically in the paper. That code wasn't giving the expected results and my job was to find why and make it work.

The mostly challenging part was to understand the maths and the physics behind those equations. Wavelengths, Simpson Integration, matrices, I had to understand those notions to be able to handle the code.

LI *et al.* PHYS. REV. APPLIED 2, 064002 (2014)

Here, the wave number along the y direction can be expressed as $k_{yn} = n\pi/(l+d)$ $(n = 0, 1, 2, \ldots)$ and $k_{xn} = \sqrt{k_0^2 - k_{yn}^2}$. The symbols A^+, A^-, C^+, and C^- denote the coefficients in the thin channel, and B_n^+ and B_n^- denote the coefficients of the nth mode with the normal wave number k_n. Note that the geometry size along the y direction of regions represented as A and C (viz., d) is much smaller compared to the working wavelength. It is, therefore, possible in this case to simplify the analytical derivation by considering the plane-wave component only; i.e., the summations over n have not applied in Eqs. (5), (6), (9), and (10). To obtain more accurate results for the regions represented as B whose geometrical size (viz., $l + d$) is comparable to a_y, the summation of normal modes is employed.

Considering that the continuity of pressure should be satisfied on the boundary between region A and region B at $x = -w$ and the boundary between regions B and C at $x = -w - d$, a transfer matrix can be constructed to connect the coefficients between regions A and C as follows:

$$\begin{bmatrix} C^+ \\ C^- \end{bmatrix} = M_1 \begin{bmatrix} A^+ \\ A^- \end{bmatrix}, \tag{12}$$

where

$$M_1 = \begin{bmatrix} \frac{\beta^2-\alpha^2+1}{2\beta} & -\frac{\beta^2-(\alpha+1)^2}{2\beta}e^{-ik_0w} \\ \frac{\beta^2-(\alpha-1)^2}{2\beta}e^{ik_0w} & -\frac{\beta^2-\alpha^2+1}{2\beta} \end{bmatrix}, \tag{13}$$

where

$$\alpha = \sum_{n=0}^{\infty} \frac{k_0 d}{k_{xn}(l+d)} \frac{1+e^{i2k_{xn}d}}{1-e^{i2k_{xn}d}} \Phi_n^1 \Phi_n^1,$$
$$\beta = \sum_{n=0}^{\infty} \frac{k_0 d}{k_{xn}(l+d)} \frac{2e^{ik_{xn}d}}{1-e^{i2k_{xn}d}} \Phi_n^1 \Phi_n^2, \tag{14}$$

with

$$\Phi_n^1 = \frac{1}{d}\int_l^{l+d} \phi_n(y)dy, \quad \Phi_n^2 = \frac{1}{d}\int_{l+l}^{l+l+d} \phi_n(y)dy. \tag{15}$$

It can be found that the transfer matrix from regions C to E is the same as the above matrix. Therefore, the coefficients in region I can be expressed as

$$\begin{bmatrix} I^+ \\ I^- \end{bmatrix} = M_1^4 \begin{bmatrix} A^+ \\ A^- \end{bmatrix}. \tag{16}$$

Considering the continuity of pressure and volume velocity at the interface between regions I and J, and the fact that the left boundary of region J is a hard boundary, the coefficients in region I can be expressed as

$$\frac{I^+}{I^-} = -\frac{1+\alpha}{1-\alpha} e^{ik_0w}. \tag{17}$$

The pressure field in the incident region (viz., $x \geq 0$ and $0 \leq y \leq a_y$) can be expressed as

$$p(x,y) = p_i + p_r = e^{-ik_0x} + \sum_{n=0}^{\infty} R_n \psi_n(y) e^{ik'_{xn}x}, \tag{18}$$

where $\psi_n(r)$ is the transverse eigenmode

$$\psi_n(y) = \sqrt{2-\delta_{0n}}\cos(k'_{yn}y), \tag{19}$$

with the wave number along the y direction given as $k'_{yn} = n\pi/(l+d+2t)$ and $k'_{xn} = \sqrt{k^2 - k'^2_{yn}}$. Following the same procedure, the transfer matrix connecting the incident region and region A is

$$\begin{bmatrix} A^+ \\ A^- \end{bmatrix} = M_2 \begin{bmatrix} 1 \\ R_0 \end{bmatrix}, \tag{20}$$

with

$$M_2 = \begin{bmatrix} \left(1-\frac{\beta'}{2}\right)e^{-ik_0w} & \frac{\beta'}{2}e^{-ik_0w} \\ 1+\frac{\alpha'}{2} & -\frac{\alpha'}{2} \end{bmatrix}, \tag{21}$$

with

$$\alpha' = \frac{a_y}{d} - \sum_{n=0}^{\infty} \frac{k_0}{k'_{xn}}(\Psi'_n)^2, \quad \beta' = \frac{a_y}{d} + \sum_{n=0}^{\infty} \frac{k_0}{k'_{xn}}(\Psi'_n)^2. \tag{22}$$

Combining Eqs. (16), (17), and (20), the total transfer matrix can be expressed as

$$R_0 = -\frac{(1-\alpha)m_{11} + (1+\alpha)e^{ik_0w}m_{21}}{(1-\alpha)m_{12} + (1+\alpha)e^{ik_0w}m_{22}}, \tag{23}$$

with

$$M = M_1^4 M_2 = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}. \tag{24}$$

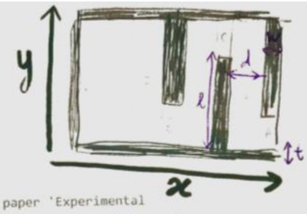
C. Realization of the full control of reflected waves

Once the complex reflection coefficient is obtained, the impedance of the presented metasurface can be expressed as

$$Z_s = \frac{1-R_0}{1+R_0}. \tag{25}$$

Then the admittance of each labyrinthine unit can be expressed as $\bar{\beta} = \rho_0 c_0 / Z_s = i\tan(\phi/2)$ with $\phi = \arg(R_0)$ being the phase shifts induced by each labyrinthine unit. It

064002-4




```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

'''
This code models and references symbols from the paper 'Experimental
Realization of Full Control of Reflected Waves with Subwavelength Acoustic
Metasurfaces' by Yong Li et al.

The symbols t, d, l and w refer to the dimensions of the labyrinthian passage,
which is subfigure C in Figure 1:
    t: Outer width
    d: Channel width
    l: ~~Channel~~ Bar length
    w: Bar width

All measurements are in metres.
'''

class AcousticModeler(object):
    def __init__(self, freq, t, d, l, w, simpsonIterations=None, sumIterations=None):
        if simpsonIterations == None:
            simpsonIterations = 100

        if sumIterations == None:
            sumIterations = 1000

        self.t = float(t)
        self.d = float(d)
        self.l = float(l)
        self.w = float(w)
        self.freq = float(freq)
        self.simpsonIterations = simpsonIterations
        self.sumIterations = sumIterations

        wlen = 343.0 / self.freq
        self.K0 = (2.0 * math.pi) / wlen

        self.computeMatrix()

    # Top of page 4
    def kyn(self, n):
        return (float(n) * math.pi) / (self.l + self.d)
```

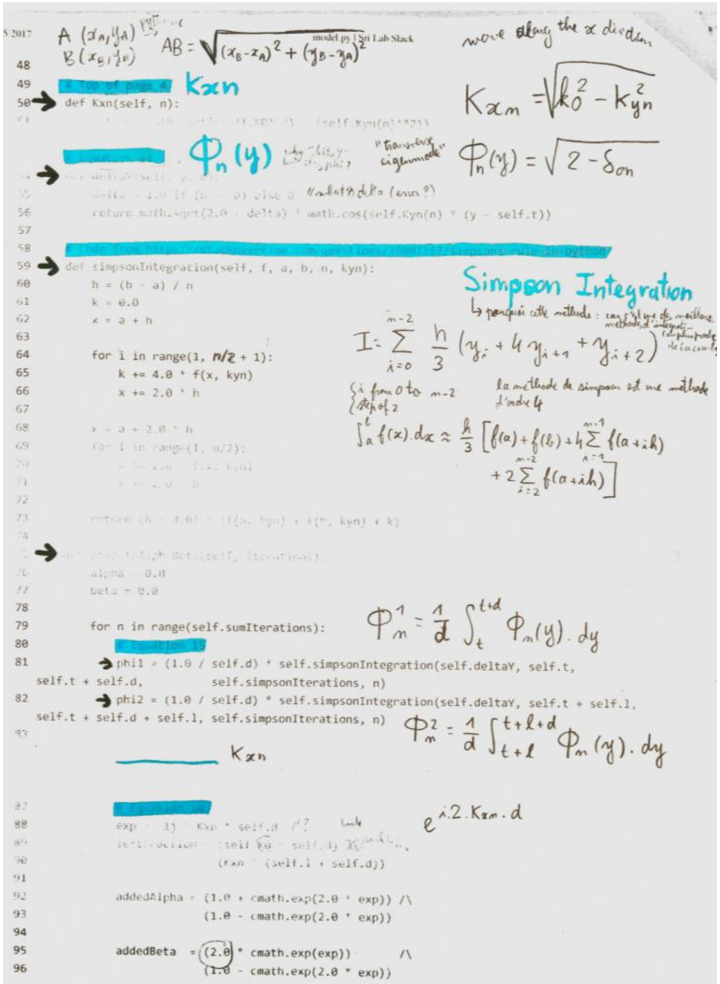
$\lambda = \frac{343}{f}$

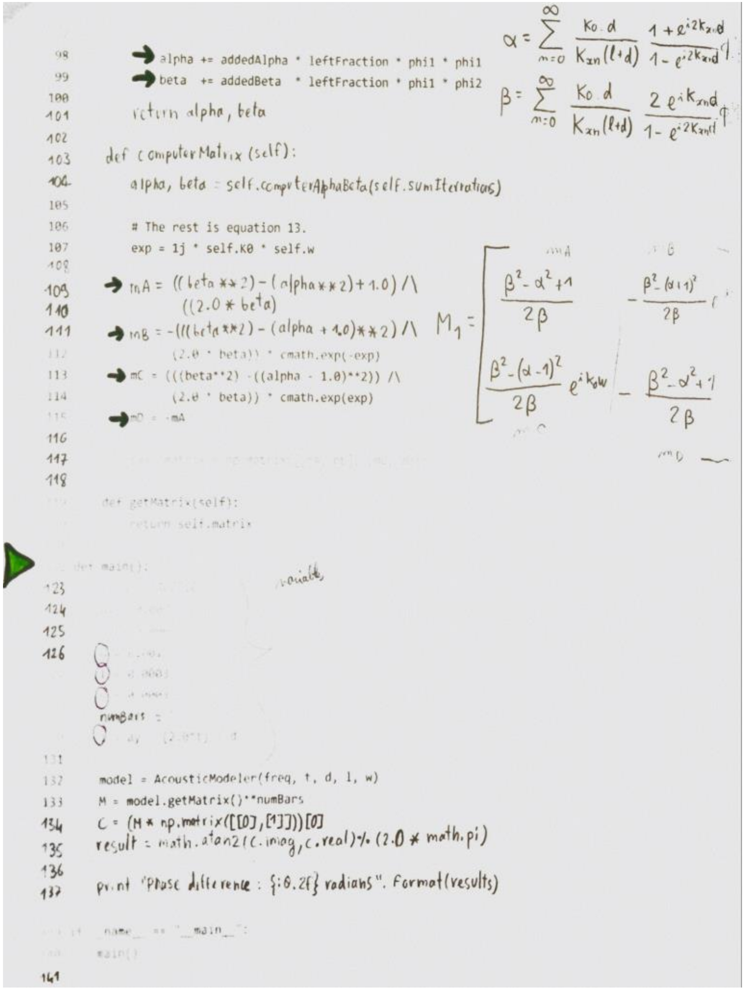
$K_0 = \frac{2\pi}{\lambda}$

wave along the y direction

Kyn

$K_{yn} = \frac{n\pi}{(l+d)}$





Blue page : <u>Analitical methods and formulas used to calculate the wavelength (page 4 of the metasurfaces article [1])</u>

3 white pages : <u>First version of the python code</u>

Organizing data

To check if that code were giving the correct answers, I had to organize the datas of the bricks parameters and the corresponding phase angle that I was supposed to get.

Tables below :

ARTICLE [1]

[1] Yong Li, Xue Jiang, Rui-qi Li, Bin Liang, Xin-ye Zou, Lei-lei Yin, and Jian-chun Cheng1, *Experimental Realization of Full Control of Reflected Waves with Subwavelength Acoustic Metasurfaces*, PHYSICAL REVIEW

fequency (Htz)	3430	values in mm								
wavelegth (meters) λ0	0,10000	#	1	2	3	4	5	6	7	8
nb of bars		m	3	2	2	2	2	1	1	2
nb of bars		n	2	2	1	1	1	1	1	1
brick height λ0/8	12,500	x	12,500	12,500	12,500	12,500	12,500	12,500	12,500	12,500
brick width λ0/8	12,500	y	12,500	12,500	12,500	12,500	12,500	12,500	12,500	12,500
brick depth ?		z								
outer width (from graph p2)		t	1,4	2,4	1,3	1,8	2,1	0,1	1,6	4
channel width d=x/(m+n)-w		d	1,5	2,125	3,1667	3,1667	3,1667	5,25	5,25	3,1667
bar length l=y-2t-d		l	8,2	5,575	6,7333	5,7333	5,1333	7,05	4,05	1,3333
bar width	1	w	1	1	1	1	1	1	1	1
phase delay (from graph p2)			0	0,25	0,5	0,75	1	1,25	1,5	1,75
actual phase angles (degree)			**0**	**45**	**90**	**135**	**180**	**225**	**270**	**315**
code test_v2 phase angles (degree)			360	331,5	0	261,09	75,334	318,85	328,81	91,82
code testsimpson_v5.2 angles (degree)			277,75	269,51	286,93	272,51	268,27	229,12	23,437	293,18

ARTICLE [2]

[2] Gianluca Memoli, Mihai Caleap, Michihiro Asakawa , Deepak R. Sahoo , Bruce W. Drinkwater & Sriram Subramanian, *Metamaterial bricks and quantization of meta-surfaces*, Nature Communications 8, 14608, (2017)

fequency (Htz)	40000	values are in mm																
wavelegth (meters) λ0	0,00858	#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
nb of bars	2	m	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
nb of bars	2	n	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
brick height λ0	0,00858	x	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750	8,5750
brick width λ0/2	0,00429	y	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875
brick depth λ0/2	0,00429	z	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875	4,2875
outer width λ0/40	0,00021	t	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144	0,2144
channel width		d	0	0,216	0,212	0,207	0,189	0,161	0,166	0,171	0,134	0,257	0,234	0,230	0,207	0,203	0,175	0,152
bar length		l	0	0,062	0,092	0,112	0,132	0,152	0,162	0,171	0,191	0,221	0,241	0,251	0,271	0,281	0,301	0,321
bar width λ0/20	0,00043	w	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288	0,4288
fillet radius																		
target phase angles (degree)			**0**	22,5	**45**	67,5	**90**	112,5	**135**	157,5	**180**	202,5	**225**	247,5	**270**	292,5	**315**	337,5
actual phase angles (degree)			0	23,3	47,5	67,4	89,7	115,6	134,4	159,3	177,3	204,8	226,2	246,4	271,4	295,3	315	335,3
code test_v1 phase angles (degree)			0	183,31	183,25	183,18	182,91	182,48	182,56	182,64	182,07	183,92	183,58	183,52	183,18	183,12	182,7	182,35

Debugging part (python, excel)

Python

After several trials, that code didn't gave the expected results.
Like the code I made was based on a beginning of someone else's code, certain parts were a bit blurry (the Simpson integration part and the matrices), so I decided to restart it from scratch by constructing the functions steps by steps.

STEP ①

First a simple function first $ax + b$:

```python
import numpy as np

#https://stackoverflow.com/questions/16001157/simpso
def integrationSimpson(integrand,lower,upper,*args):
    panels = 100000 #iterations
    limits = [lower, upper] #interval
    h = ( limits[1] - limits[0] ) / (2 * panels)
    n = (2 * panels) + 1
    x = np.linspace(limits[0],limits[1],n)
    y = integrand(x,*args)


    I = 0
    start = -2
    for looper in range(0,panels):
        start += 2 #step of two
        counter = 0
        for looper in range(start, start+3):
            counter += 1
            if (counter ==1 or counter == 3):
                I += ((h/3) * y[looper])
            else:
                I += ((h/3) * 4 * y[looper])
    return I

def f(x,a,b):
    return a * x + b

I = integrationSimpson(f,0,1,1,0)
print(I)
```

STEP ②

Then replace $ax + b$ by $\cos(ax) + b$ with a by $\sqrt{{K_0}^2 - {K_{y_n}}^2}$:

```python
def f(x,a,b):
    return (np.cos(a * x) + b)
```

STEP ③

Then by modifying the result of the square root :

If the result of $\sqrt{{K_0}^2 - {K_{y_n}}^2} < 0 \Rightarrow \frac{e^{-(ax+b)} + e^{(ax+b)}}{2}$

if the result of $\sqrt{{K_0}^2 - {K_{y_n}}^2} > 0 \Rightarrow \cos(ax + b)$

```python
def f(x,a,b):
    freq = 3430. # article [1]
    wlen = 343. / freq
    K0 = (2. * math.pi) / wlen
    # values parameters article [1]
    ax = (wlen / 8.)
    ay = (wlen / 8.)
    w = .001 # 1mm
    t = 0.0014 # VARIABLE BRIQUE 1
    numBars = 5. # VARIABLE BRIQUE 1
    d = (ax/numBars) - w
    l = ay - (2.*t) - d

    # loop (not clean /!\ to update)
    n = A
    Kyn = (n * np.pi) / (l + d)
    restemp = ((K0) ** 2 ) - ((Kyn) ** 2)
    a = A

    if restemp > 0:
        restemp = restemp
        a = math.sqrt(restemp)
        finalres = np.cos(a * x) + b
    else:
        restemp = - restemp
        a = math.sqrt(restemp)
        e1 = (np.exp(-(a * x + b)))
        e2 = (np.exp(+(a * x + b)))
        finalres = ((e1 + e2) /2)
    restemp = finalres

    return finalres #np.cos(a * x) + b

#=========================================
for A in range(0,10):
    I = SimpsonIntegration(f,0,.001,A,1) #SimpsonIntegration(f,
    print(I) #FINAL RESULT
```

STEP ④

Then by getting closer and reach the final *Psi* function $\phi_n(y) = \sqrt{2 - \delta_{0n}}\,.\cos[K_{y_n}(y - t)]$

STEP ⑤

Integrating the *Psi* function :

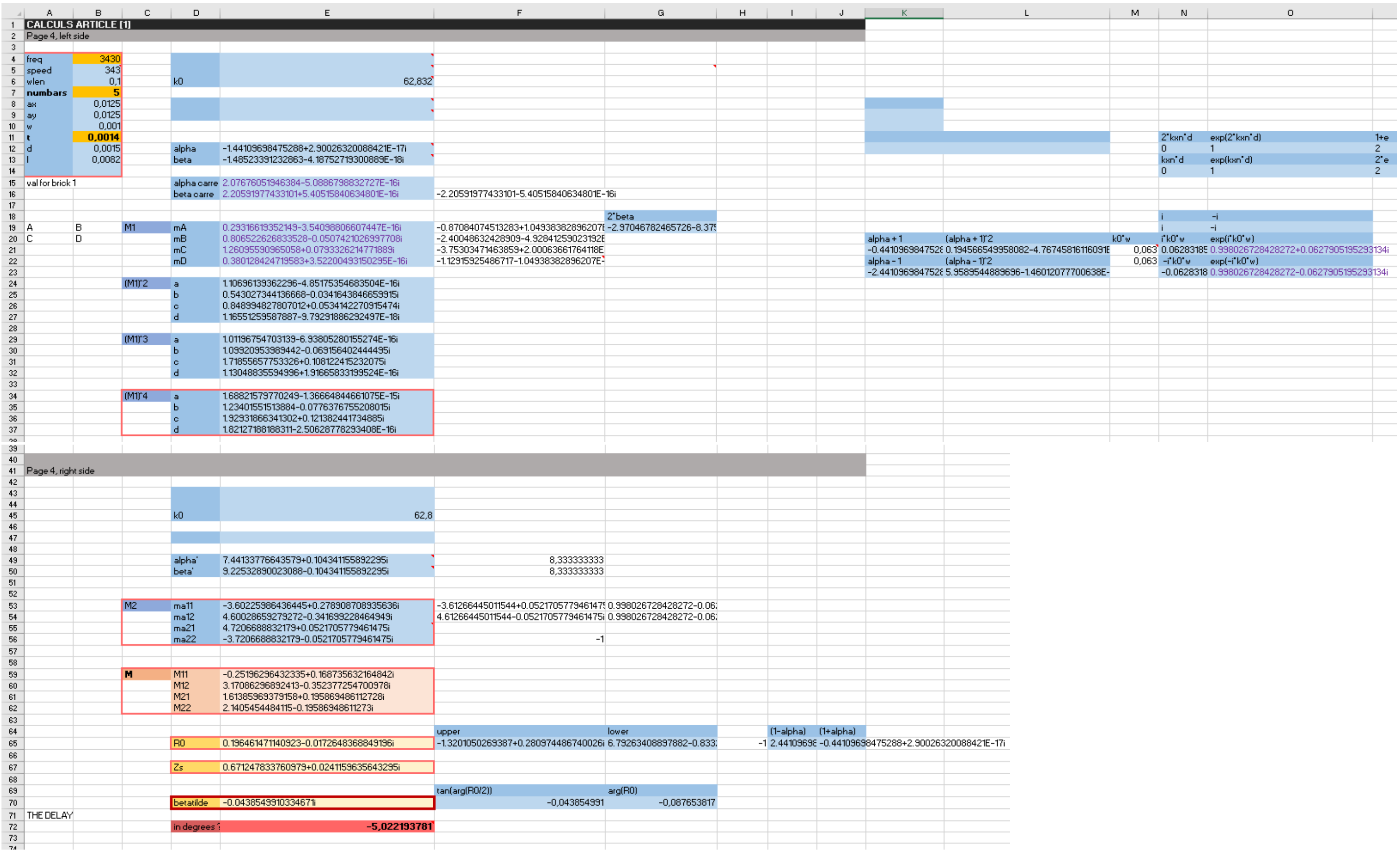
$\phi_n^1 = \frac{1}{d}\int_t^{t+d} \phi_n(y).dy$ and $\phi_n^2 = \frac{1}{d}\int_{t+l}^{t+l+d} \phi_n(y).dy$

(see code in Appendix)

Excel

In that second python code, the results (of the phase delay after going through the brick) given weren't the expected ones either.

We wanted to be able to see all the details of every operation that were made to detect any error in the Python code, so we decided to make the same operations as in the Python code but with an excel file.
There we realized that we had forgotten an important part in the operation: some operations were with complex numbers but we neglected that part before.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	CALCULS ARTICLE [1]															
2	Page 4, left side															
4	freq	3430														
5	speed	343														
6	wlen	0,1		k0	62,832											
7	numbars	5														
8	ax	0,0125														
9	ay	0,0125														
10	w	0,001														
11	t	0,0014												2*kxn*d	exp(2*kxn*d)	1+e
12	d	0,0015		alpha	-1.44109698475288+2.90026320088421E-17i									0	1	2
13	l	0,0082		beta	-1.48523391232863-4.18752719300889E-18i									kxn*d	exp(kxn*d)	2*e
14														0	1	2
15	val for brick 1			alpha carre	2.07676051946384-5.0886798832727E-16i											
16				beta carre	2.20591977433101+5.40515840634801E-16i	-2.20591977433101-5.40515840634801E-16i										
18							2*beta							i	-i	
19	A	B	M1	mA	0.29316619352149-3.54098806607447E-16i	-0.87084074513283+1.04938382896207E	-2.97046782465726-8.375							i	-i	
20	C	D		mB	0.806522626833528-0.0507421026997708i	-2.40048632428909-4.92841259023192E					alpha + 1	(alpha + 1)^2	k0*w	i*k0*w	exp(i*k0*w)	
21				mC	1.26095590965058+0.0793326214771889i	-3.75303471463859+2.00063661764118E					-0.44109698475288	0.194566549958082-4.76745816116091E	0,063	0.06283185	0.998026728428272+0.0627905195293134i	
22				mD	0.380128424719583+3.52200493150295E-16i	-1.12915925486717-1.04938382896207E-					alpha - 1	(alpha - 1)^2	0,063	-i*k0*w	exp(-i*k0*w)	
23											-2.44109698475288	5.9589544889696-1.46012077700638E-		-0.0628318	0.998026728428272-0.0627905195293134i	
24			(M1)^2	a	1.10696139362296-4.85175354683504E-16i											
25				b	0.543027344136668-0.0341643846659915i											
26				c	0.848994827807012+0.0534142270915474i											
27				d	1.16551259587887-9.79291886292497E-18i											
29			(M1)^3	a	1.01196754703139-6.93805280155274E-16i											
30				b	1.09920953989442-0.069156402444495i											
31				c	1.71855657753326+0.108122415232075i											
32				d	1.13048835594996+1.91665833199524E-16i											
34			(M1)^4	a	1.68821579770249-1.36664844661075E-15i											
35				b	1.23401551513884-0.0776376755208015i											
36				c	1.92931866341302+0.121382441734885i											
37				d	1.82127188188311-2.50628778293408E-16i											
41	Page 4, right side															
45				k0	62,8											
49				alpha'	7.44133776643579+0.104341155892295i	8,333333333										
50				beta'	9.22532890023088-0.104341155892295i	8,333333333											
53			M2	ma11	-3.60225986436445+0.278908708935636i	-3.61266445011544+0.0521705779461475	0.998026728428272-0.06									
54				ma12	4.60028659279272-0.341699228464949i	4.61266445011544-0.0521705779461475i	0.998026728428272-0.06									
55				ma21	4.7206688832179+0.0521705779461475i											
56				ma22	-3.7206688832179-0.0521705779461475i	-1										
59			M	M11	-0.25196296432335+0.168735632164842i											
60				M12	3.17086296892413-0.352377254700978i											
61				M21	1.61385969379158+0.195869486112728i											
62				M22	2.1405454484115-0.19586948611273i											
64						upper	lower		(1-alpha)	(1+alpha)						
65				R0	0.196461471140923-0.0172648368849196i	-1.3201050269387+0.280974486740026i	6.79263408897882-0.833	-1	2.4410969	-0.44109698475288+2.90026320088421E-17i						
67				Zs	0.671247833760979+0.0241159635643295i											
69						tan(arg(R0/2))	arg(R0)									
70				betatilde	-0.0438549910334671i	-0,043854991	-0,087653817									
71	THE DELAY															
72				in degrees ?	-5,022193781											

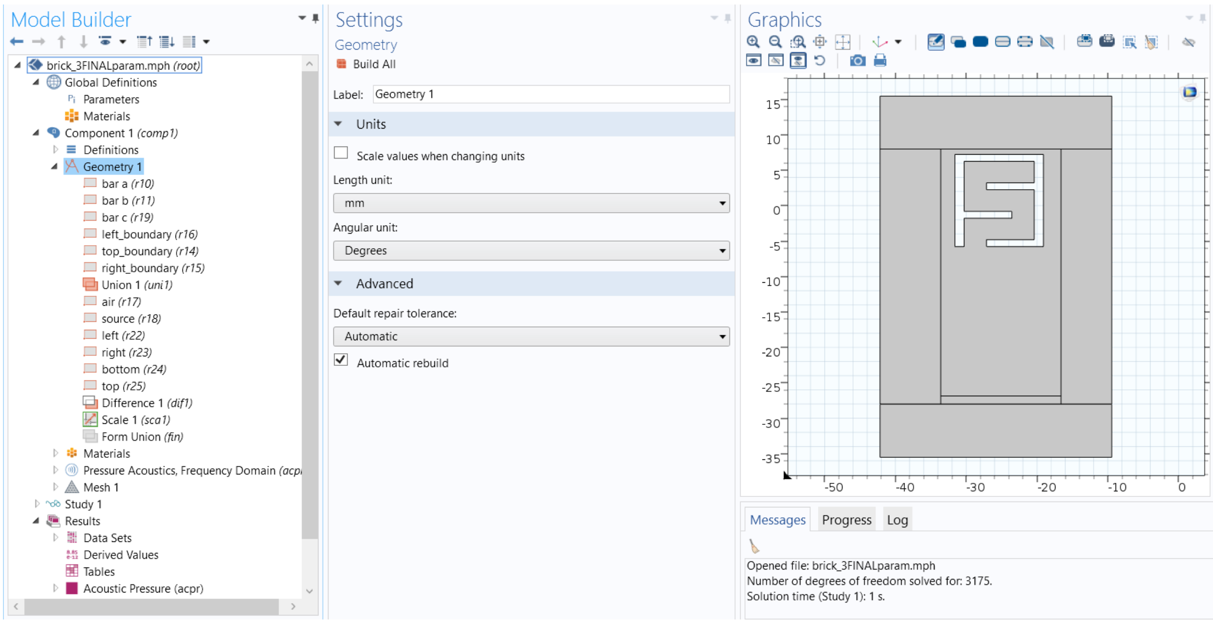
Unfortunately, those excel file result weren't the expected ones even though we checked as many details as possible. So, we were blocked with both ways (python and excel), that's why we decided to go for another way which is COMSOL.

Modelisation part (COMSOL and Java)

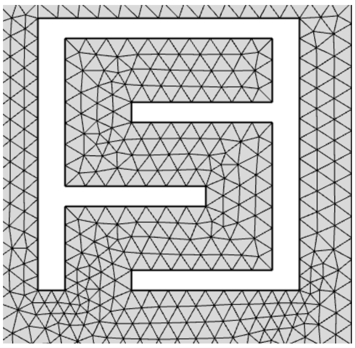
COMSOL is a software used to do physics simulations (electrical, mechanical, fluid, and chemical applications). Learning to use it wasn't an easy part because it's quite complex and with a lot of functionalities and menus. The goal here as well was to get similar results as in the article [1] by reproducing the bricks, simulate the wavelength and calculating the delay.

Creating a brick (from the article [1])

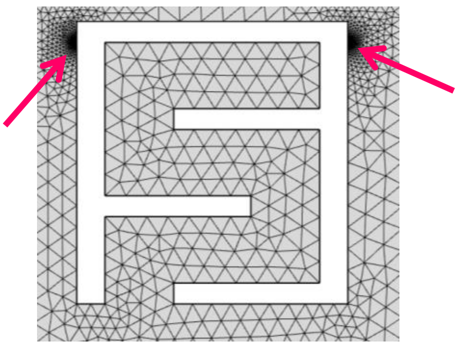
To create an object in COMSOL and be able to extract results, there is few steps to follow.
First building the geometry of the brick with parameter integration and precise measures.



Then, setting a the mesh to cover the surface to study.



Correct mesh



Incorrect mesh

Changing the parameters of bricks through a java file

In the comsol software, parameters need to be changed by clicking on them in the user interface. Changing parameters permit to go from one brick to another by going in the dedicated COMSOL section and changing them. An interesting thing with COMSOL is that files can be saved in .java file format. In these files, the parameters are in a variable that appears in the first lines of the code and be modified from there.

It was more simple form an upcoming deep learning approach to have access to .java files

It was easy to get theses .java files from the COMSOL file (with the "save as …" functionality), but COMSOL couldn't load them, once the file in the .java form. Some preliminary steps were required.

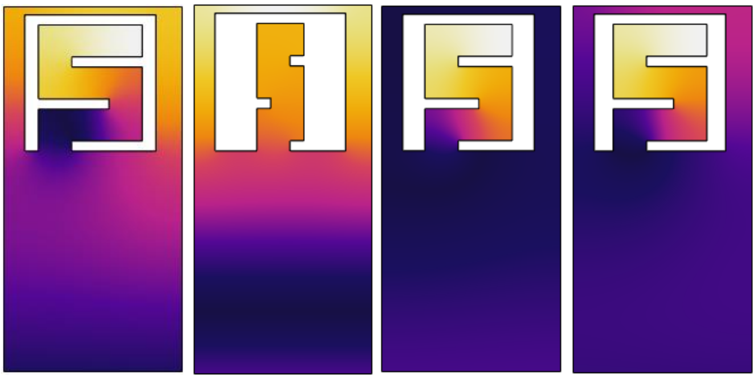
STEP ① get the .class file by compiling on the command prompt

```
comsolcompile -jdkroot "C:\Program Files (x86)\Java\jdk1.7.0_55" brickname.java
```

STEP ② execute the file

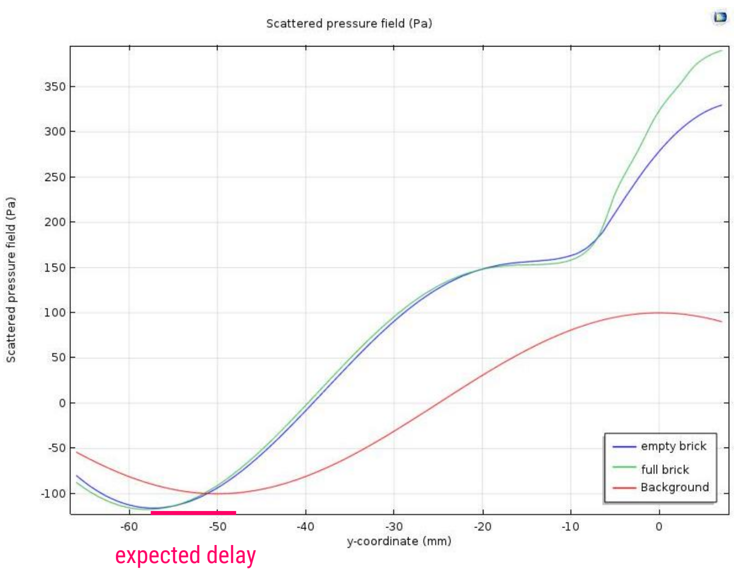
```
"C:\Program Files\COMSOL\COMSOL52a\Multiphysics\bin\win64\comsolbatch" -inputfile
"C:\Users\utilisateur\Documents\COURS\4.IUT INFORMATIQUE
A.S\STAGE\COMSOL\NiceBricksModels\brickname.class"
```

Brick n3 (manually created), bricks n8, n5 and n4 are created by using the file of brick n3 (below in that order)



Measure the pressure field

Some option in COMSOL permit to measure the pressure field amount a line. With the help of my supervisor, I succeed to obtain the expected pressure field.



Conclusion

I had the real opportunity to work on a wide range of various missions during this internship : resume and complement an existing program to find out why it did not give the expected results, implement a python program from scratch, discover and handle a new modeling software and be the first one in the lab to use it through commands lines and with java files.

I benefited from a benevolent and stimulating guidance.
The working environment between the university research laboratory and startup linked devices was particularly interesting because of the access to a rich range of advanced technologies and colleagues from different technical and geographical backgrounds.

I was able to gather computer science, cognitive science and mathematics by using of my knowledges from this year of IUT informatique année spéciale and my degree in maths applied to the cognitive sciences.

I learned to reuse the code made by another person, increased my knowledge of Python, learned the basis of a modeling software, and work on a daily “long-term” project.

Finally, the whole internship session took place in Sussex University, England, it helps me improve my english by exchange with the team and having daily conversations. If I think I am now comfortable speaking in English but drafting a report or an essay is still quite complicated.

Appendix

VIDEO : “MetamaterialbricksandquantizationofMeta-surfaces.mov”

CODES

Simpson Integration “testsimpson_v5.2.py”

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
'''
This code models and references symbols from the paper 'Experimental
Realization of Full Control of Reflected Waves with Subwavelength Acoustic
Metasurfaces' by Yong Li et al.

The symbols t, d, l and w refer to the dimensions of the labyrinthian passage,
which is subfigure C in Figure 1:
    t: Outer width
    d: Channel width
    l: Bar length
    w: Bar width

    ax : brick height
    ay : brick width

All measurements are in metres.

Tallulah GILLIARD
t.gilliard@gmail.com
INTERACT Lab, Sussex University
'''

import cmath
import math
import numpy as np

#///////////////////////////////////////////////////////////////////////////
class Transmission(object):
    def __init__(self, freq, t, d, l, w, ax, ay, n, wlen=None, A=None):
        #======== VARIABLES <
        self.freq = float(freq)
        self.wlen = float(wlen) if (wlen != None) else 343. / self.freq
        self.K0 = (2. * math.pi) / self.wlen
        self.n = float(n)

        self.ay = float(ay)
        self.ax = float(ax)
        self.t = float(t)
        self.d = float(d)
        self.l = float(l)
        self.w = float(w)

        self.matrix1 = self.Matrix1()
        self.matrix2 = self.Matrix2()

        self.A = A if (A != None) else 10
        #======== VARIABLES >


#PAGE 4 LEFT SIDE //////////////////////////////////////////////////////
    #======================================================================
    # page 4, up-left
    def Kyn1(self, n):
        #self.n = self.A # based on A loops
        Kyn1 = (float(self.n) * np.pi) / (self.l + self.d)
        return Kyn1

    #======================================================================
    # page 3, equation 11
    def phi_eigenmode(self, x, n, b):

        restemp = ((self.K0) ** 2 ) - ((self.Kyn1(n)) ** 2)
        #a = self.A
        self.b = self.t # b correspond to the outer width

        #======== delta <
        if n == 0:
            delta=math.sqrt(2)
        else:
            delta = 1
        #======== delta >

        #======== PHI <
        if restemp > 0: # positive case
            restemp = restemp
            a = math.sqrt(restemp)
            phinal = delta*np.cos(a * (x - b))
        else: # negative case
            restemp = - restemp
            a = math.sqrt(restemp)
            e1 = (np.exp(-(a * (x - b))))
            e2 = (np.exp(+(a * (x - b))))
            phinal = delta*((e1 + e2) /2)
        restemp = phinal
        #======== PHI >

        return phinal

    #======================================================================
    #https://stackoverflow.com/questions/16001157/simpsons-rule-in-python
    def SimpsonIntegration(self, integrand, lower, upper, *args):
        #SimpsonIntegration(self, f, lower, upper, a, b)
        panels = 100000 #iterations (number of rectangles)
        limits = [lower, upper] #interval
        h = ( limits[1] - limits[0] ) / (2 * panels)
        n = (2 * panels) + 1
        x = np.linspace(limits[0],limits[1],n)
        y = integrand(x,*args)

        I = 0
        start = -2
        for looper in range(0,panels):
            start += 2 #step of two
            counter = 0
            for looper in range(start, start+3):
                counter += 1
                if (counter ==1 or counter == 3):
                    I += ((h/3) * y[looper])
                else:
                    I += ((h/3) * 4 * y[looper])
        return I

    #======================================================================
    # page 4, equation 14
    def alphabeta1(self):
        alpha = 0.
        beta = 0.

        for n in range(0,10): #self.A
            try:
                # equation 15
                phi1 = (1 / self.d) * self.SimpsonIntegration
                (self.phi_eigenmode, self.t,         self.t + self.d, n, self.t)
                #SimpsonIntegration(f,lower,upper,a,b)
                phi2 = (1 / self.d) * self.SimpsonIntegration
                (self.phi_eigenmode, self.t + self.l,self.t + self.l + self.d, n, self.t)
                #SimpsonIntegration(f,lower,upper,a,b)

                # Top of page 4
                Kxn1 = cmath.sqrt((self.K0**2) - (self.Kyn1(n)**2))

                # Equation 14
                exp =  1j * Kxn1 * self.d
                leftFraction = (self.K0 * self.d) /\
                (Kxn1 * (self.l + self.d))

                addedAlpha = (1. + cmath.exp(2. * exp)) /\
                (1. - cmath.exp(2. * exp))

                addedBeta  = (2. * cmath.exp(exp))       /\
                (1. - cmath.exp(2. * exp))

                alpha += addedAlpha * leftFraction * phi1 * phi1
                beta  += addedBeta  * leftFraction * phi1 * phi2
            except ZeroDivisionError:
                pass
        alpha1 = alpha
        beta1 = beta
        return alpha1,beta1

    #======================================================================
    # equation 13
    def Matrix1(self):
        alpha, beta = self.alphabeta1() #!!! ?

        # The rest is equation 13.
        exp = 1j * self.K0 * self.w

        mA = ((beta**2) - (alpha**2) + 1.) /\
             (2. * beta)
        mB =-(((beta**2) - (alpha + 1.)**2)  /\
              (2. * beta)) * cmath.exp(-exp)
        mC = (((beta**2) -((alpha - 1.)**2)) /\
              (2. * beta)) * cmath.exp(exp)
        mD = -mA

        matrix1 = np.matrix([[mA, mB], [mC, mD]])
        return matrix1
```

```python
#PAGE 4 RIGHT SIDE ///////////////////////////////////////////////////////////////

    #============================================================================
    # Between equation 19 and 20
    def Kyn2(self,n):
        #self.n = self.A # based on A loops
        Kyn2 = (float(self.n) * np.pi) / (self.l + self.d + (self.t * 2.))
        return Kyn2

    #============================================================================
    # Between equation 19 and 20
    def Kxn2(self, n):
        return cmath.sqrt((self.K0**2) - (self.Kyn2(n)**2))

    #============================================================================
    # Equation 19
    def psi_eigenmode(self, x, n):

        restemp2 = ((self.K0) ** 2 ) - ((self.Kyn2(n)) ** 2)


        #======== delta <
        if n == 0:
            delta=math.sqrt(2)
        else:
            delta = 1
        #======== delta >

        #======== PSI <
        if restemp2 > 0: # positive case
            restemp = restemp2
            a = math.sqrt(restemp)
            psinal = delta*np.cos(a * x)
        else: # negative case
            restemp = - restemp2
            a = math.sqrt(restemp)
            #print("a :",a)
            #print("x :",x)
            e1 = (np.exp(-(a * x)))
            #print("e1 :",e1)
            e2 = (np.exp(+(a * x)))
            #print("e2 :",e2)
            psinal = delta*((e1 + e2) /2)
        restemp = psinal
        #======== PSI >

        return psinal

    #===========================================================================
    # Equation 22
    def alphabeta2(self):
        summed = 0.

        for n in range(0,10): #!!!
            try:
                psiprime = (1 / self.d) * self.SimpsonIntegration
                (self.psi_eigenmode, self.t,         self.t + self.d, n)
                #SimpsonIntegration(f,lower,upper,a)
                summed += (self.K0 / self.Kxn2(n)) * (psiprime**2)
            except ZeroDivisionError:
                pass

        alpha = (self.ay / self.d) - summed
        beta  = (self.ay / self.d) + summed

        alpha2 = alpha
        beta2 = beta
        return alpha2, beta2

    #===========================================================================
    # Equation 21
    def Matrix2(self):
        alpha, beta = self.alphabeta2()

        dummy = 1j * self.K0 * self.w

        m11 = (1. - (beta/2.)) * cmath.exp(-dummy)
        m12 = (beta/2.) * cmath.exp(-dummy)
        m21 = 1. + (alpha/2.)
        m22 = - (alpha/2.)

        matrix2 = np.matrix([[m11, m12], [m21, m22]])
        return matrix2

    #============================================================================
    # Equation 23
    def R0(self):
        M = self.FinalMatrix(self.matrix1, self.matrix2)
        alpha, beta = self.alphabeta1()

        m11 = M.item((0,0))
        m12 = M.item((0,1))
        m21 = M.item((1,0))
        m22 = M.item((1,1))

        exp = 1j * self.K0 * self.d
        return - ((1. - alpha) * m11) + ((1. + alpha) * cmath.exp(exp) * m21) /\
                 ((1. - alpha) * m12) + ((1. + alpha) * cmath.exp(exp) * m22)

    #============================================================================
    # RESULTS

    # Equation 24
    def FinalMatrix(self, matrix1, matrix2):
        return (matrix1**4) * matrix2

    # impedance
    def getImpedance(self):
        return (1. - self.R0) / (1. + self.R0)

    # phase
    def getPhase(self):
        R0 = self.R0()
        return math.atan2(R0.real, R0.imag) + math.pi


#//////////////////////////////////////////////////////////////////////////////
def main():
        #======== VALUES <
        print("===============RESULTS===============")
        n = 10 # ???
        freq = 3430. # article [1]
        wlen = 343. / freq # !!! répétition
        #A=10
        numBars = 5. # VARIABLE BRIQUE 1
        ax = (wlen / 8.)
        ay = (wlen / 8.)
        w = .001 # 1mm (fixed)
        t = 0.0014 # VARIABLE BRIQUE 1
        d = (ax/numBars) - w
        l = ay - (2.*t) - d
        #======== VALUES >

        model = Transmission(0, t, d, l, w, ax, ay, n, wlen=wlen, A=10)
        result = model.getPhase()

        #in radians
        print ("Delay in radians :")
        print (result)
        #in degrees
        print("Delay in degrees :")
        print (math.degrees(result))

if __name__ == "__main__":
    main()
```

Glimpse of the code “brick_4FINALparam.java“ from the COMSOL saving

```java
/*
 * brick_4FINALparam.java
 */

import com.comsol.model.*;
import com.comsol.model.util.*;
/** Model exported on Aug 14 2017, 10:42 by COMSOL 5.2.1.152. */
public class brick_4FINALparam {

  public static Model run() {
    Model model = ModelUtil.create("Model");

    model
         .modelPath("C:\\Users\\utilisateur\\Documents\\COURS\\4.IUT INFORMATIQUE A.S\\STAGE\\COMSOL\\NiceBricksModels");

    model.label("brick_4FINALparam.mph");

    model.comments("Untitled\n\n");

    model.param().set("t", "1.8 [mm]", "outer width");
    model.param().set("l", "5.733 [mm]", "bar length");
    model.param().set("ax", "12.5 [mm]", "brick height");
    model.param().set("ay", "12.5 [mm]", "brick width");
    model.param().set("w", "1 [mm]", "bar width");
    model.param().set("d", "3.1667 [mm]", "channel width");
    model.param().set("Ox", "-31.6", "originX (bottom left point)");
    model.param().set("Oy", "-5", "originY (bottom left point)");

    model.modelNode().create("comp1");

    model.geom().create("geom1", 2);

    model.mesh().create("mesh1", "geom1");

    model.geom("geom1").lengthUnit("mm");
    model.geom("geom1").create("r10", "Rectangle");
    model.geom("geom1").feature("r10").label("bar a");
    model.geom("geom1").feature("r10").set("size", new String[]{"l", "w"});
    model.geom("geom1").feature("r10").set("pos", new String[]{"(Ox+ax)-t-l", "Oy"});
    model.geom("geom1").create("r11", "Rectangle");
    model.geom("geom1").feature("r11").label("bar b");
    model.geom("geom1").feature("r11").set("size", new String[]{"l", "w"});
    model.geom("geom1").feature("r11").set("pos", new String[]{"Ox+t", "Oy+(w+d)"});
    model.geom("geom1").create("r19", "Rectangle");
    model.geom("geom1").feature("r19").label("bar c");
    model.geom("geom1").feature("r19").set("size", new String[]{"l", "w"});
    model.geom("geom1").feature("r19").set("pos", new String[]{"(Ox+ax)-t-l", "Oy+((2*w)+(2*d))"});
    model.geom("geom1").create("r14", "Rectangle");
    model.geom("geom1").feature("r14").label("top_boundary");
    model.geom("geom1").feature("r14").set("size", new String[]{"ax", "w"});
    model.geom("geom1").feature("r14").set("pos", new String[]{"Ox", "Oy+((3*w)+(3*d))"});
    model.geom("geom1").create("r15", "Rectangle");
```

References

PAPERS

[1] Yong Li, Xue Jiang, Rui-qi Li, Bin Liang, Xin-ye Zou, Lei-lei Yin, and Jian-chun Cheng1, *Experimental Realization of Full Control of Reflected Waves with Subwavelength Acoustic Metasurfaces*, PHYSICAL REVIEW APPLIED **2**, 064002 (2014)

[2] Gianluca Memoli , Mihai Caleap, Michihiro Asakawa , Deepak R. Sahoo , Bruce W. Drinkwater & Sriram Subramanian, *Metamaterial bricks and quantization of meta-surfaces*, Nature Communications **8**, 14608, (2017)

[3] Dennis Li, Lucian Zigoneanu, Bogdan-Ioan Popa, and Steven A. Cummer, *Design of an acoustic metamaterial lens using genetic algorithms*, Acoustical Society of America **132**, 2823 (2012)

METAMATERIALS

Video about metamaterials
https://youtu.be/BcVxRyvipcU
INTERACT Lab website
http://interact-lab.com
High transmission acoustic focusing by impedance-matched acoustic meta-surfaces
http://repository.kaust.edu.sa/kaust/bitstream/10754/594725/1/1.4939932.pdf

COMSOL to Java video
https://youtu.be/5DRNpCn-GY0
COMSOL Multiphysics guide
https://extras.csc.fi/math/comsol/3.4/doc/multiphysics/guide.pdf
COMSOL Java API Reference Guide
https://lost-contact.mit.edu/afs/ict.kth.se/pkg/comsol/4.3/doc/pdf/mph/COMSOLJavaAPIReferenceGuide.pdf

OTHERS

Sensation templates
https://developer.ultrahaptics.com/knowledgebase/the-first-10-sensation-templates/
Simpson integration
http://www.emathhelp.net/notes/calculus-2/numerical-approximate-integration/simpsons-rule/
Deeplearning guide
http://www.deeplearningbook.org/