Politechnical University of Valencia
University of Valencia

Master's Thesis

Signal fitting using Genetic Algorithms

Santiago Pérez Romero

supervised by
LLuís Miquel García Raffi
José Manuel Calabuig Rodríguez

Abstract

In this paper is dealt the classical curve fitting problem, specifically with acoustic signals. The pattern of this type of signals is analyzed. As a non-linear problem we have chosen one non-linear method to solve it. Genetic algorithm is the method selected, which can give us full freedom at the time of find solutions. It is shown the design and configuration of the algorithm. At the end is tried with different data, and are explained some problems and how can be dealt.

Contents

1 Introduction

Curve fitting is a known problem in engineering and science. It consist in finding an approximated compact representation from a given data set. The literature about this topic is extensive, with papers since the second half of the past century [1] (1959). Automatic curve fitting is an alive research subject as shows, for example, the reference [2] form three years ago. Roughly speaking the objective of curve fitting is to find a set of functions which span approaches the original function. If we only consider the problem for finding the coefficients for the linear combination of functions from the set that minimize the error, the problem has a well-known solution. But in most situations, the functions from the set can depend on parameters in a non-linear manner in such way that our problem consists in both determine the coefficients of the span and the values of the parameters of the functions in order to minimize the difference between the original data and the representation. Fitting is used in many situation as simulation or statistical inference. The first challenge is modeling data, is necessary to analyze how is our dataset and what we want to get of it. Because the useful information depends on the data and our objectives. For example, for a research about arrhythmia using the signal of an electrocardiogram it could be more important to get the frequency than the power. In this study we dealt with an specific signals, but using a general method, which would be useful in other environment focusing in the modeling of the peaks. The motivation of this research is to develop and study a general tool to get an accurate information from a signal without the support of an expert.

1.1 Context

Our data set correspond to acoustic signals. Active control of sound and vibration is a relatively new and fast growing field of research and application. The control of low-frequency noise and vibration has traditionally been difficult and expensive. The passive control of noise makes necessary the use of large mufflers, enclosures for noise control and isolation systems and/ or structural damping treatment (vibration absorbers) for vibration control. The idea of using active sound cancellation as an alternative to passive control of sound was first proposed in the 30's of the past century. The basic idea is to use an acoustic/vibrational controlled source to introduce a secondary (control) disturbance into the system to cancel the existing (primary) disturbance, thus resulting in an attenuation of the original sound/vibration. The cancelling disturbance was to be derived electronically based upon a measurement of the primary disturbance. The identification of the frequencies composing the primary disturbance is fundamental for the effectiveness of the method.[3]

Then, the processing of this type of data is important. Of course, peak identification and fitting have applications also in, for example, music, nuclear spectroscopic or mechanics. As we can see in [8] this type of data is composed by peaks with different height and width. Calculate the parameters of that peaks is fundamental to recognize some physical characteristic of the system analyzed.

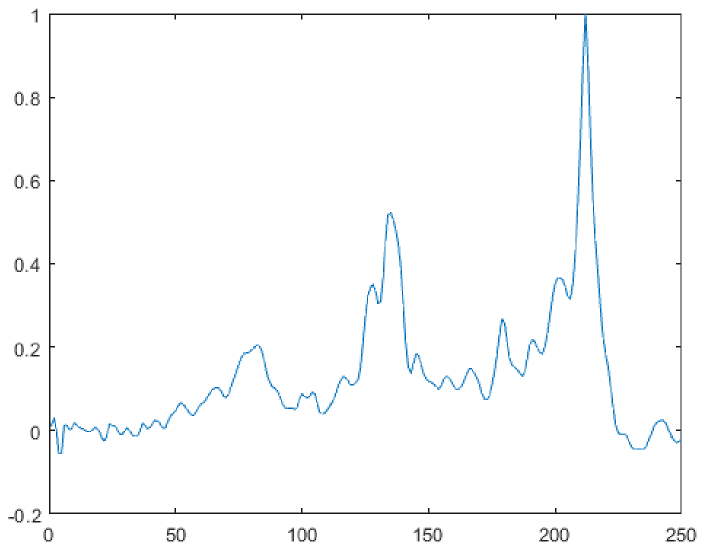


Figure 1: An example of acoustic signal

1.2 Objective

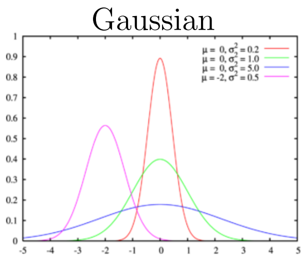
Our objective is to design an algorithm which is going to give us these parameters of the signal, that basically consists of determining both the number of peaks contained in the spectrum and the relevant characteristics of them from the point of view of frequency identification: the centroid and the width of the peak are the most relevant parameters (there are others as the asymmetry, the existence of queues,...).

1.3 Problem

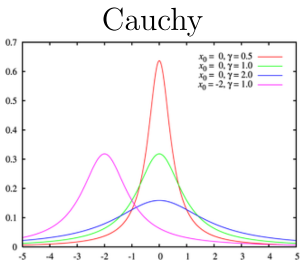
This type of problem consist in solving the equations system

$F(x_1) = S(x_1)$
$F(x_2) = S(x_2)$
$F(x_3) = S(x_3)$
...
$F(x_n) = S(x_n)$

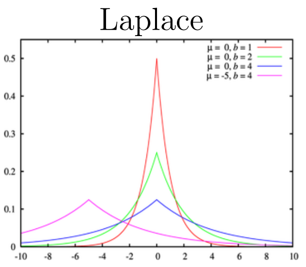
where **n** is the size of the data, **S** the signal and **F** the objective function. For modeling this function, we consider that it can be expressed as a combination of other functions that represent the peaks. For modeling peaks we are going to consider the next three functions.



$$f(x) = a * e^{-\frac{(\mu-x)^2}{2\sigma^2}}$$



$$f(x) = \frac{1}{\pi}\frac{\gamma}{(x-\mu)^2+\gamma^2}$$



$$f(x) = \frac{1}{2a} * e^{-\frac{|x-\mu|}{a}}$$

The reason to use different types of peaks is to consider different shapes of them but as we will see later gaussian shape is a very good approach in most situations. This system is not lineal because variables appears in the argument of a function which is not linear. Nonlinear systems are difficult to solve. Can be approximated by linear equations (linearization) but in this work we will use an optimization technique Genetic Algorithms (GA). GA belongs to the larger class of evolutionary algorithms (EA).

1.4 Genetic Algorithms

For solving the system we propose genetic algorithm. Genetic Algorithm is a technique whose main ideas are inspired by the process of natural selection. Thanks to this can achieve complex objectives that traditional search tools cannot. The first reference we have about genetic algorithms was from John Holland by the University of Michigan in 1970. Since then, this tool had been used until our days in science, design and engineering. The main idea of GA is that the organisms that are more capable will survive while the others will disappear.

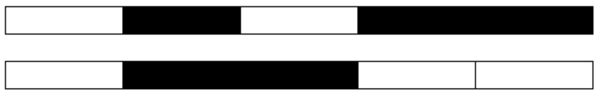


Figure 2: Initial population

Here we have two solutions designing crosswalks. Can been imagined as organisms that are evolving. First population can be made randomly or using another methods, we know is improbable that one of them to be the best solution or at least a good solution. We model the crosswalk as a vector of 1's and 0's, 1 for black and 0 for white. But with a simple genetic algorithm we can get it. First of all we need is a fitness function, is the function which score the solutions. In this case, $\mathbf{f(x)} = \sum_{\mathbf{i}} |(\mathbf{x_i} - \mathbf{x_{i-1}})|$. That means that every change of color over the crosswalk give it 1 point.

That means first solution is a better crosswalk. Now is the time of the evolution.
The evolution of solutions consist basically in two methods, mutation and crossover. The

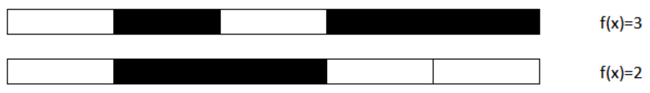


Figure 3: Initial population scored with fitting function

mutation process consist in change randomly some characteristic of a gen, allowing the algorithm to get values that can't be gotten with the initial population value, compared with crossover mutation process try to reach the nearer local maximum. The crossover operator combine two, or more, solutions randomly in a new one. Represents the sexual reproduction, the "child" has characteristics of both "fathers", it allows to "jump" to another part of the function, avoiding to stay in the same local maximum.

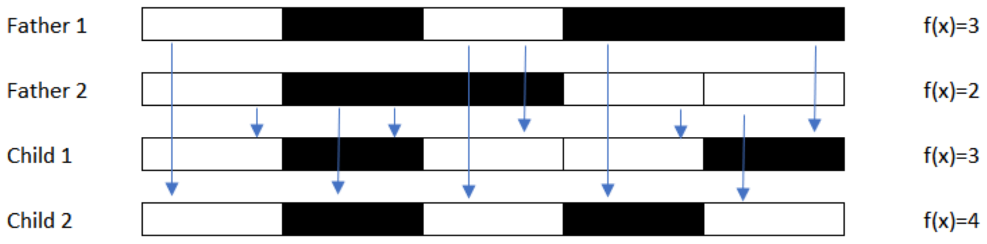


Figure 4: Crossover process

There it can seen an example of crossover, this is an educational prove, actually in a generation the number of child with lower fitting value compared with them father is higher. Mutation and crossover are stochastic process. Both change randomly the organisms, this brings a certain instability to the technique, which will be corrected with the selection process.
The selection process choose which elements are gonna survive to the next generation and which will be deleted. The test must not be so elitist, because the key in genetic algorithm is the diversity of solutions and reject similar solutions although they have good fitting value. Usually are selected a little number of top solutions, mutations from them and the crossover between all the population. Although the selection strategy depends on the problem as all the other configurations (mutation, crossover) it can be set by default, because it only affect notoriously to computation time. In a new generation all the organisms are not gotten thanks to mutation or crossover, some individuals jump to the next generation without any change. It use to be called the elite, a minimal proportion of the population with the best scores in the fitting function. Is important to save a few of these organisms but not too many to maintain diversity.

GA needs stopping criteria, which will select when stop the process. Stops condition depends of the design. We can finish the process after a certain amount of time or number of generations, or after some generations without find any better individual. When GA stops, it can returns the best solution or entire generations. In this case we care about the best individual.

2 Design

2.1 Gen format

For the problem we are dealing with we must first design a representation of the elements. To simplify programming every organism are gonna to be a vector of real numbers.
$\phi = \{\mathbf{a_1 a_2 a_3 .....a_n}\}$
The first one is the baseline of the signal. The others are parameters of all the functions which compose the signal in groups of 4, hence the number of variables is $(\mathbf{number\_of\_peaks} * \mathbf{4}) + \mathbf{1}$. In this group of 4 parameters the first we have is the control parameter which selects if there exists or not one peak at that position in the signal and the type of it (Gaussian, Laplace, Cauchy). So, for the peak **n**:

- if $a_{4n+1} < 0$ then the peak **n** is rejected.
- if $0 \leq a_{4n+1} < 0.25$ then the peak **n** is modelled with Gaussian function.
- if $0.25 \leq a_{4n+1} \leq 0.5$ then the peak **n** is modelled with Cauchy function.
- if $a_{4n+1} > 0.5$ then the peak **n** is modelled with Laplace function.

The remaining three are peak parameters **n**:

- **Gaussian**
  a_{4n+2} = height of the peak
  a_{4n+3} = position of the peak
  a_{4n+4} = width of the peak
- **Cauchy**
  a_{4n+2} = half-width at half-maximum
  a_{4n+3} = position of the peak
  a_{4n+4} = nothing
- **Laplace**
  a_{4n+2} = scale parameter
  a_{4n+3} = position of the peak
  a_{4n+4} = nothing

It is possible to use any type of function and different baseline functions raising both the number of variables and the computation time. But tests show us that the use of Gaussian functions for representing peaks is a good candidate for the shape of peaks in the acoustic case.

2.2 Fitting function

For evaluate if one solution is better than other, we have to calculate the error between the solution proposed and the original signal. For doing that, we calculate the mean quadratic error which looks useful for this purpose in [1][2][6][7][8][10].

$$\mathbf{f(X)} = \frac{\sum_{\mathbf{i}}(\mathbf{X_i}-\mathbf{Y_i})^{\mathbf{2}}}{\mathbf{n}}$$

where $\mathbf{X}$ is the evaluation at points x_i of the approximation of signal obtained with our procedure, $\mathbf{Y}$ the original signal and $\mathbf{n}$ the length of both signals that must be equal.
We use this function for penalizing big errors at some specific point. In the search of general method the signal will be normalized to values between 0 and 1 allowing to use the same stopping criteria, explained before, for inputs with different scales.

2.3 Elitism, crossover and mutation

For tests we use different setups. Although it doesn't change the results however computation time changes. We let the default Matlab configuration for crossover and mutation. About elitism we save the 30% better gens of all generations.

2.4 Population

The size of the populations is one of the most important parameter in genetic algorithm. The larger the population, the easier to explore the search space. But the time required by a GA (complexity) to converge is $\mathcal{O}(n \log(n))$ being n the population size. If the number of individuals is too high, too many fitness function evaluations are necessary and the CPU time is too high. A small population can result in the algorithm being trapped at a local minimum. For this purpose we put empirically this function for calculate it:

$$\mathbf{f(x)} = \mathbf{min}([\mathbf{max}([\mathbf{50}, \mathbf{4} * \mathbf{numberOfVariables}]), \mathbf{300}])$$

It works well with the size of data used, but with data having more peaks, the size of the population calculated with this formula is not enough.

2.5 Initialization

GA starts with an initial population of N potential solutions randomly generated, each one being called individual. We saw that this generation is the most important. If is formed totally random is gonna be hard to get and optimal solution in a affordable interval of time. On another hand, is needed to put the maximum number of peaks. It should be bigger than the number of possible real peaks in the signal, to let the algorithm select which peaks are going to be left. For this reason, we analyze the signal to identify all the peaks that compose it. Calculating the derivative to be zero we can get the position and the height of the peak. Also, we can find the hypothetical baseline, finding the minimum value of the signal and subtracting it to all peaks' height.

2.6 Algorithm

For the algorithm we used the Optimization Toolbox of Matlab, which has a function called ga which has a lot of parameters for been configured. Also we used Parallel Toolbox that allow us to use multi-core characteristic.

2.7 Stopping criteria

GA is an iterative method that requires a stop condition. The stopping criteria in our procedure depends of the number of variables are in the system. When new genes do not improve more than $10^{-}6$ the value of the fitting function over a certain number of generations the process is terminated. That number is calculated by the following function:

$$\mathbf{f(x) = max([10, 40 - (numberOfVariables/10)])}$$

It is a decision for performance, giving more freedom to simple problems.

2.8 Output

When the algorithm stops, the output we get is the gen, or solution, that give us the parameters of the peaks that compose the approached signal, being it the closer to the original in terms of the error evaluated at points x_i . For example, consider this signal.

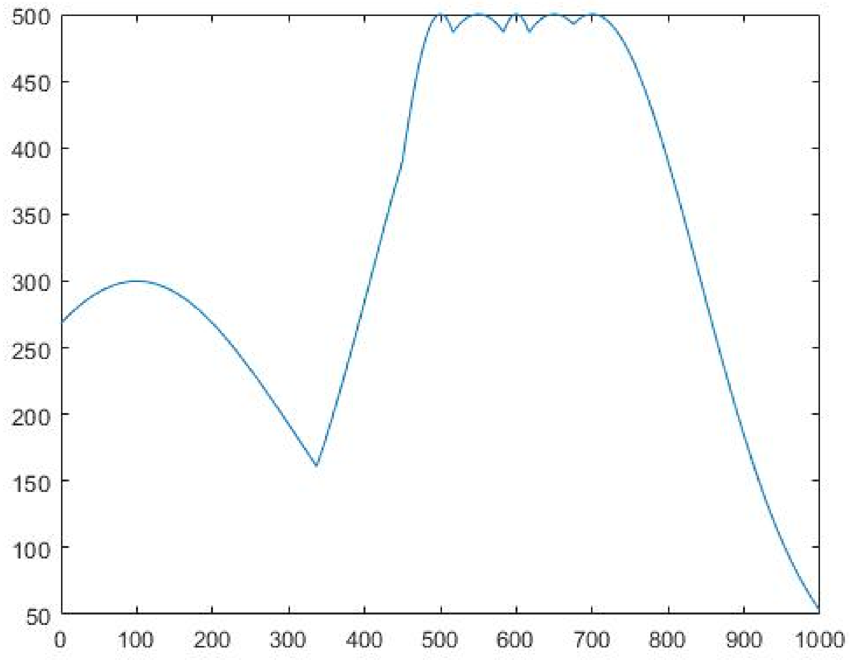


Figure 5: Artificial signal as example

And the best individual sent by the genetical algorithm is:

$$\mathbf{0.096736 + 0.49301 * e^{-\frac{x-103.6205}{2*139.8192^2}} + 0.56646 * e^{-\frac{x-489.78}{2*68.6907^2}} + 0.14288 * e^{-\frac{x-397.575}{2*112.3036^2}}}$$
$$\mathbf{-10.1828 * e^{-\frac{x-1127.7537}{2*-36.6676^2}} + 0.97027 * e^{-\frac{x-700}{2*117.5037^2}}}$$

With this visual representation:

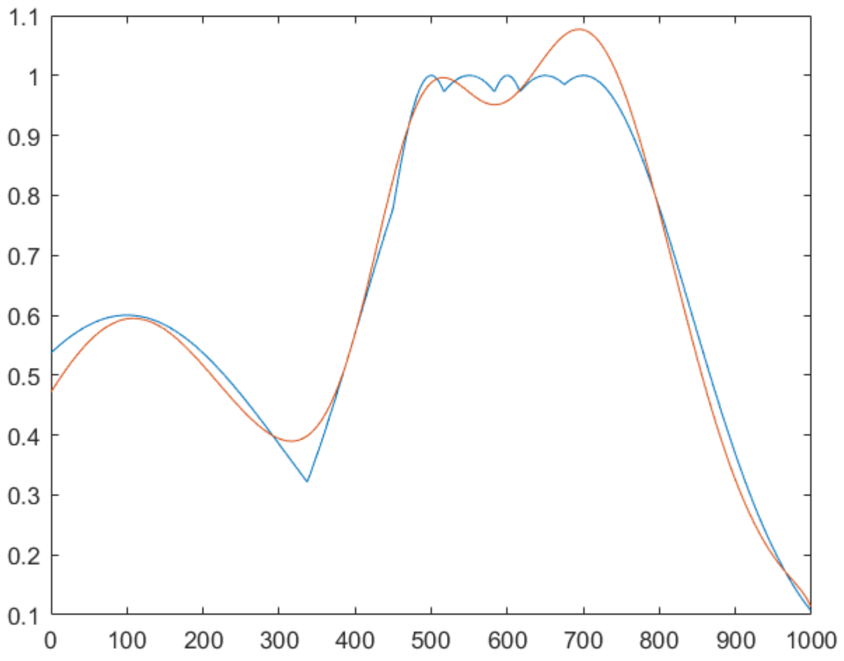


Figure 6: Signal and its fitting with GA.

with an error of **0.0011**. Analyzing the output function we know the position, height and width of the peaks. Also can be seen that are five gaussian. In this example, is shown a typical problem, there is a peak outside the range of the signal. Maybe its valleys affect the signal. This fact is analyzed later.

3 Results

In order to analyze the efficiency of our procedure, we are going to test it in different situations, starting with a signal with only one peak and finishing with true acoustic signals.

3.1 Simple peak

In this first example it is shown a gaussian.

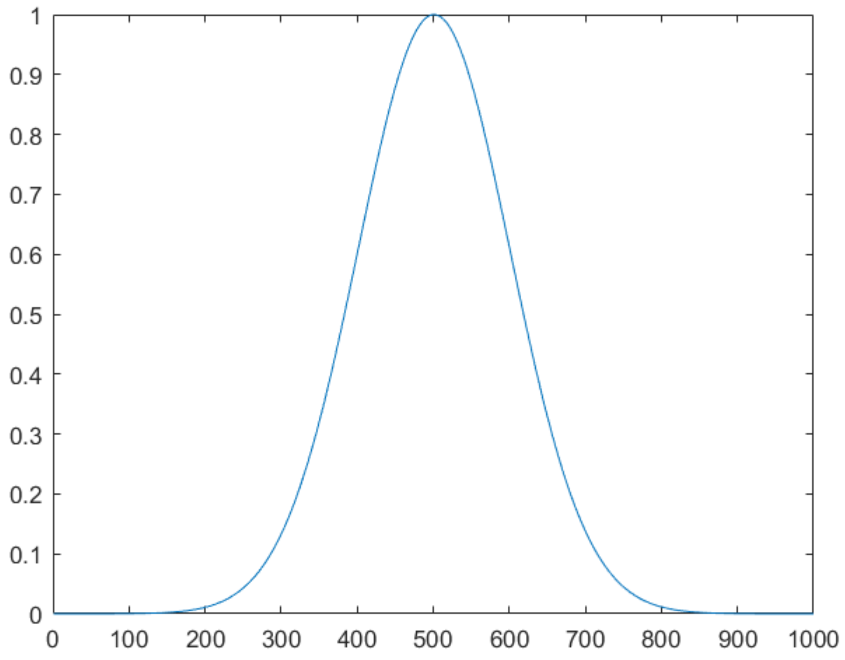


Figure 7: Simple peak plot.

This signal was performed with this function:

$$\mathbf{f(x) = 1 * e^{-\frac{x-500}{2*100^2}}}$$

We put the signal to the algorithm, but the algorithm does not know this information. The information we get at the end of the execution is this:

$$\mathbf{f(x) = 2.687e^{-7} + 1.0423 * e^{-\frac{x-501.2339}{2*97.9608^2}}}$$

The error is $\mathbf{2.3002 * e^{-4}}$. In this table is shown a study with ten executions with the same input data to show the stability with this example.

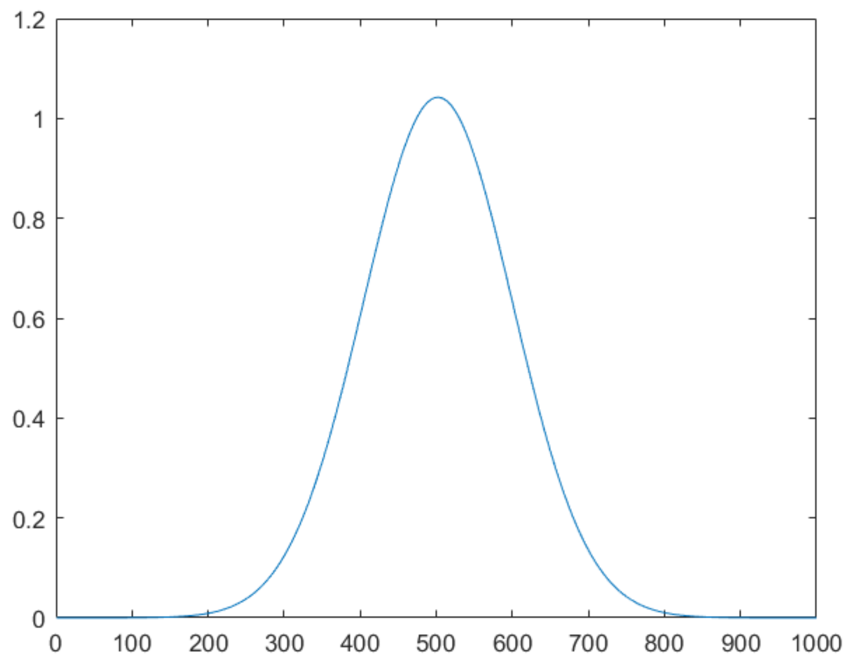


Figure 8: Fitting of simple peak.

Error	N. Peaks	Baseline	Height	Position	Width
0.0014	1.0000	0.0000	0.8931	499.7574	106.9723
0.0095	1.0000	0.0000	0.7703	500.1724	137.5096
0.0008	1.0000	0.0171	0.9116	499.9749	103.8219
0.0000	1.0000	0.0000	1.0000	500.0000	98.2661
0.0000	1.0000	0.0000	0.9931	499.9245	99.9169
0.0038	1.0000	-0.0350	1.1766	498.0713	89.8397
0.0002	1.0000	0.0000	1.0449	500.0501	97.1972
0.0002	1.0000	0.0000	0.9566	500.3340	102.9381
0.0000	1.0000	0.0000	1.0000	500.0749	99.8548
0.0002	1.0000	0.0000	1.0000	500.0000	95.8739

This data show us how accurate is the algorithm with one peak example. The next table represents mean deviation of the values.

Error	N. Peaks	Baseline	Height	Position	Width
0.0002	0	0.0000	0.0441	0.0752	3.4942

We can see that width is the value that changes more, but is possible because there is one result with a width too different from the average, the second one in the table, that show the lower height and the higher width. This will be common in the plots, caused by the stochastic nature of the genetic algorithms.

3.2 Two peaks

The second experiment use a signal with two peaks.

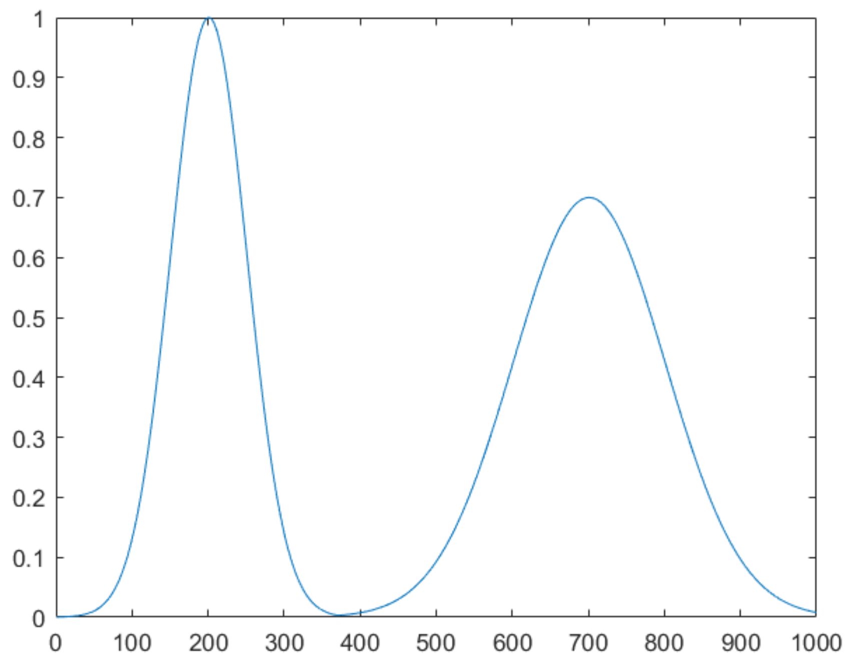


Figure 9: Artificial signal with two peaks

This signal was performed with two gaussian, but not composed, getting the maximum value between both:

$$\mathbf{f(x) = max(1 * e^{-\frac{x-200}{2*50^2}}}, 0.7 * e^{-\frac{x-700}{2*100^2}})$$

This is not normal in acoustic where different signals were added in the final signal. Although in this example it does not have notorious effect in the final signal. When we put this signal, we get for example this:

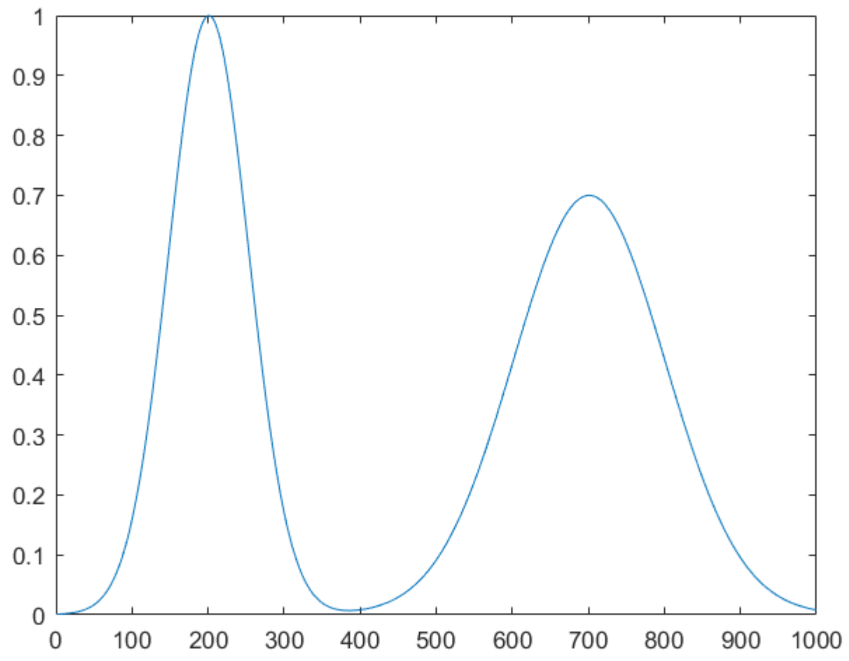


Figure 10: Signal fitting

$$\mathbf{f(x) = 0.00032943 + 0.99966 * e^{-\frac{x-200}{2*52.6096^2}} + 0.69966 * e^{-\frac{x-700}{2*99.5551^2}}}$$

The error is $\mathbf{1.8193e^{-4}}$. The stability study with ten executions:

Error	N. Peaks	Baseline	Height I	Position I	Width I	Height II	Position II	Width II
0.0000	2.0000	0.0002	0.9997	200.0000	49.0501	0.6997	700.0000	98.9859
0.0001	2.0000	0.0003	0.9997	200.0000	50.4947	0.6997	700.0000	102.7855
0.0001	2.0000	0.0002	0.9997	200.0000	48.6014	0.6997	700.0000	100.3702
0.0003	2.0000	0.0003	0.9320	200.0000	52.5507	0.6997	700.0000	102.4726
0.0005	2.0000	0.0003	0.9997	200.0000	54.1819	0.6997	700.0000	96.5676
0.0001	2.0000	0.0002	0.9997	200.0000	49.5329	0.6997	700.0000	95.8793
0.0003	2.0000	0.0001	0.9997	200.0000	51.5253	0.6591	700.0000	104.0052
0.0009	2.0000	0.0000	0.9997	200.0000	52.8569	0.6362	700.0000	112.8173
0.0001	2.0000	0.0005	0.9997	200.0000	51.3586	0.6997	700.0000	100.4397
0.0017	2.0000	0.0003	0.8561	200.0000	58.7400	0.6997	700.0000	95.4120

This data show us how accurate is the algorithm with two peaks example. The next table represents mean deviation of the values.

Error	N. Peaks	Baseline	Height I	Position I	Width I	Height II	Position II	Width II
0.0001	0	0.0001	0	0	1.6620	0	0	2.9903

This values looks so perfect, a little deviation between different executions and low error, we will see it does not happens all the time. With complex data we will find some dataset so unstable and another problems like overlapping peaks or peaks that will see in the function but will be inappreciable in the plot compared, with all the signal.

3.3 Double-peak

In this test, we force a double-peak with the same technique as we used for built the previous signal (getting the maximum), but with the peaks closer each other.

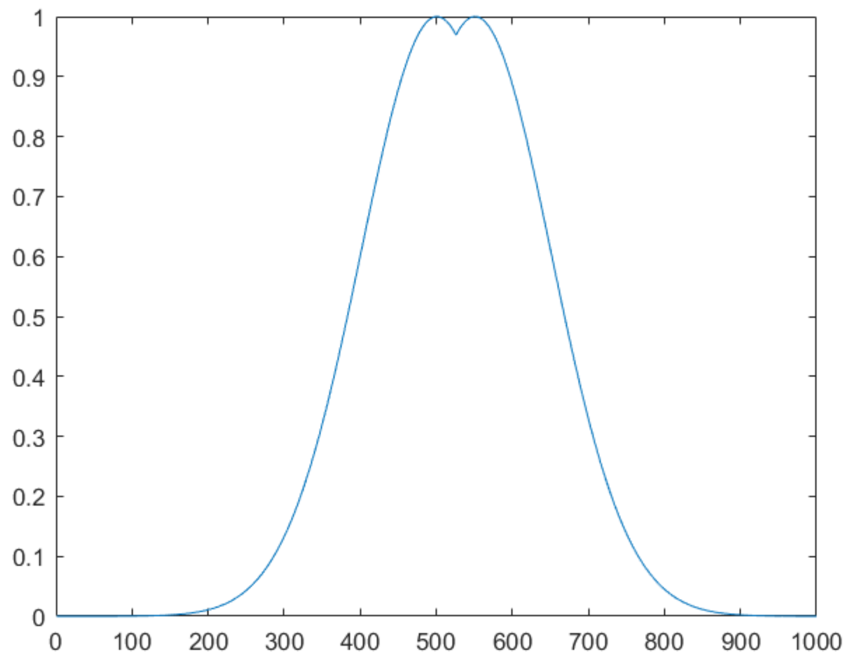


Figure 11: Artificial signal

This signal was performed with two gaussian, but not composed, getting the maximum value between both:

$$\mathbf{f(x) = max(1 * e^{-\frac{x-500}{2*100^2}}}, 1 * e^{-\frac{x-550}{2*100^2}})$$

The solution we get is:

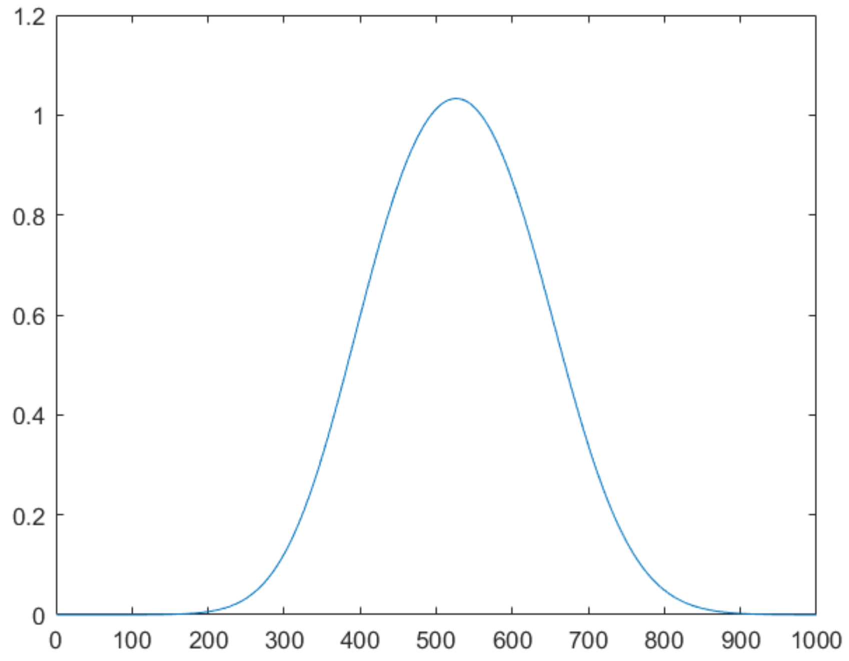


Figure 12: Signal fitting

$$\mathbf{f(x) = 1.3444e - 07 + 1.0755 * e^{-\frac{x-525.0386}{2*112.7937^2}}}$$

This is an interesting example, because the algorithm chose dataset represents only one peak. This is one of the minor objectives, we found two peaks that perform one peak with an error of $\mathbf{1.3122e^{-4}}$. It may looks and error, but in a real signal where peaks does not perform max functions as we use to create this artificial data it would be a simple peak with noise. However, the stability study shows us, that sometimes the solution considers the existence of two peaks.

Error	N. Peaks	Baseline	Height I	Position I	Width I	Height II	Position II	Width II
0.0005	1.0000	0.0000	0	0	0	1.0199	525.7247	116.7031
0.0033	2.0000	0.0000	0.6002	500.0000	135.4893	0.5869	547.5600	71.7288
0.0020	2.0000	0.0000	0.5896	574.6406	104.8461	0.6967	482.1267	88.5292
0.0026	1.0000	0.0000	0.9288	524.3041	121.7402	0	0	0
0.0005	2.0000	0.0000	1.0000	560.4279	90.2244	0.4512	412.6968	65.2211
0.0009	1.0000	0.0000	0	0	0	1.0000	527.0587	116.2537
0.0168	2.0000	0.0000	1.4074	487.6199	67.2472	0.6928	637.3801	-43.7918
0.0004	2.0000	0.0000	0.3241	498.9247	114.3668	0.7486	535.1035	111.9740
0.0020	2.0000	0.0000	0.0827	521.8876	134.2713	0.8624	526.0789	121.6984
0.0004	2.0000	0.0000	1.0000	522.8435	115.5955	0.0402	558.7550	130.1900

With this data we cannot calculate the mean deviation, and in this example it can be seen that the 70% of the solutions contain two peaks instead of one. But showing all the signal generated we realize that this is not true.

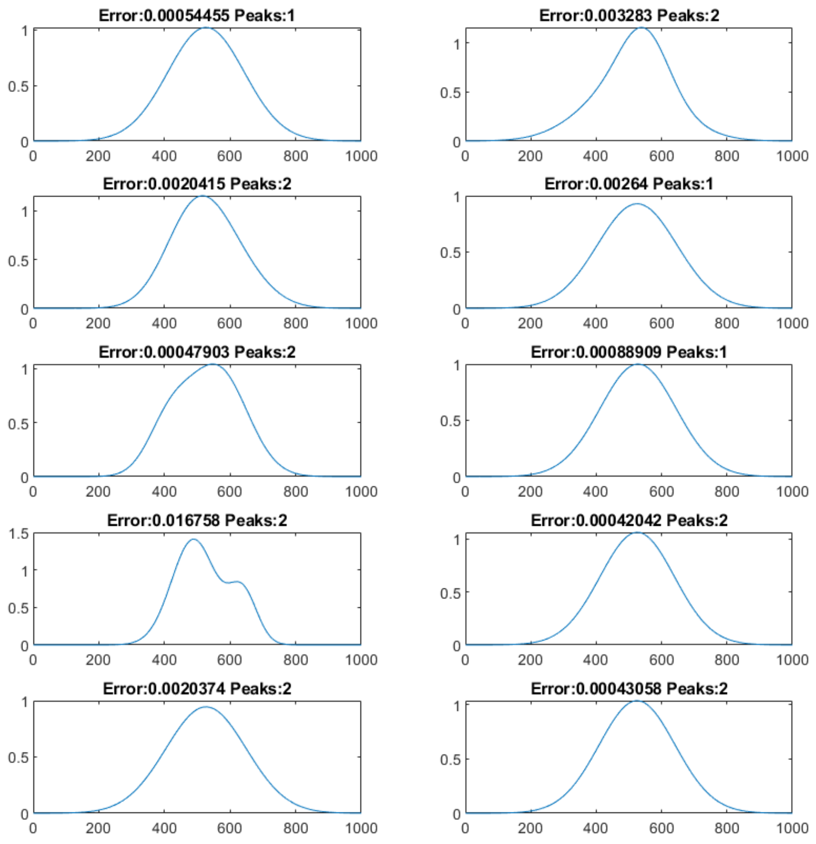


Figure 13: Plot of ten executions, showing the nature of the signal.

It is clear that the algorithm created one peak signal, but there is different ways to perform one. For example,we have two functions:

$$\mathbf{f(x) = 1 * e^{-\frac{x-500}{2*100^2}}}$$
$$\mathbf{g(x) = 0.5 * e^{-\frac{x-500}{2*100^2}} + 0.5 * e^{-\frac{x-500}{2*100^2}}}$$

But both functions plot the same function, **f** has one peak and **g** has two peaks.

In many situations, when working with signals obtained from experiments, it is very difficult to discern when it is a single peak affected by noise or doppler shift and when it is a real doublet. To distinguish a doublet, we need there exist a minimum distance between peaks. This distance can also depend on the tail of the peaks, their shape, width and can be different for each of them, etc.. In the example we have given we are going to assume that it is a single peak and we are going to carry out the stability analysis. The decision of assuming a single peak, can be automated with the data got from the stability analysis.

Error	N. Peaks	Baseline	Height	Position	Width
0.0008	1.0000	0.0000	1.0000	526.5746	118.8617
0.0008	1.0000	0.0000	1.0000	524.2061	117.4391
0.0065	1.0000	0.0000	1.2668	525.2212	102.5614
0.0008	1.0000	0.0000	1.0000	525.0993	116.9149
0.0125	1.0000	0.0000	1.3580	522.9714	97.5706
0.0020	1.0000	0.0000	1.1545	525.3737	111.8290
0.0009	1.0000	0.0000	1.0000	525.5532	115.9231
0.0077	1.0000	0.0000	0.8327	524.8910	136.7485
0.0021	1.0000	0.0000	0.9633	534.1802	122.1127
0.0026	1.0000	0.0000	0.9294	523.4025	123.1473

With this data we can calculate deviation.

Error	N. Peaks	Baseline	Height	Position	Width
0.0013	0	0.0000	0.0536	0.6735	5.1418

3.4 5 peaks

Finishing with artificial signal, we formed a signal more complex.

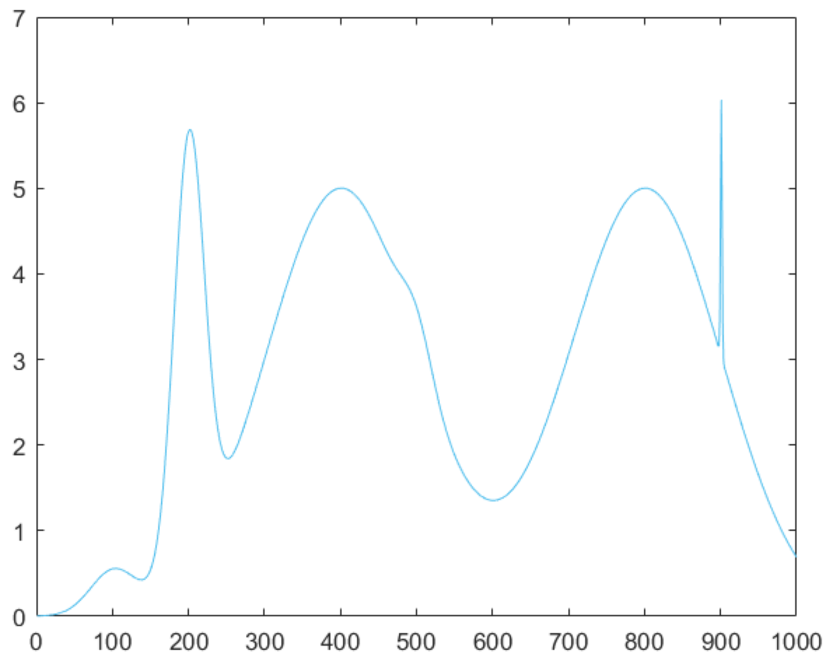


Figure 14: Artificial signal with 5 peaks

This signal was performed with this function:

$$\mathbf{f(x) = 0.5 * e^{-\frac{x-100}{2*30^2}} + 5 * e^{-\frac{x-200}{2*20^2}} + 5 * e^{-\frac{x-400}{2*100^2}} + 0.5 * e^{-\frac{x-500}{2*20^2}} + 5 * e^{-\frac{x-800}{2*100^2}} + 3 * e^{-\frac{x-900}{2*1^2}}}$$

We take the signal as input of the algorithm, but the algorithm does not know this information. The information we get at the end of the execution is this:

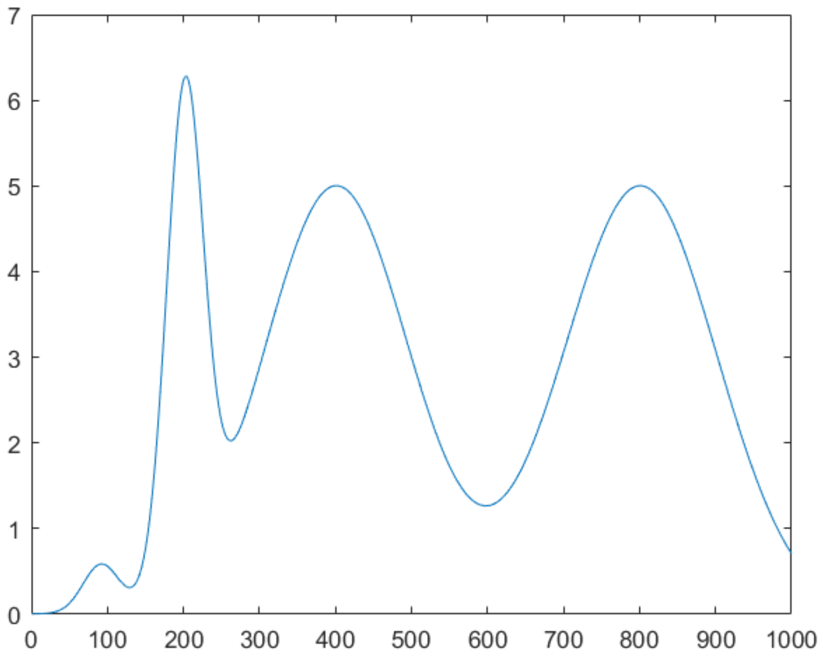


Figure 15: Signal fitting

$$f(x) = 0.0036103 + 0.55468 * e^{-\frac{x-90.4514}{2*-23.5751^2}} + 5.6822 * e^{-\frac{x-201}{2*24.206^2}} + 4.9981 * e^{-\frac{x-400}{2*96.1771^2}} + 4.9981 * e^{-\frac{x-800}{2*100.4633^2}}$$

The error is **0.0751** and the number of peaks is 4. Is important to remark that in spite of the error is the highest we have showed in all the paper, the algorithm considers the signal is formed with less peaks than the true signal, but comparing **(Fig.14)** and **(Fig.15)** looks that the missing peaks are noise. So in this case, the algorithm works as noise cleaner, although it was not originally designed for this purpose.

3.5 Real signal

The next example shows a true acoustic signal.

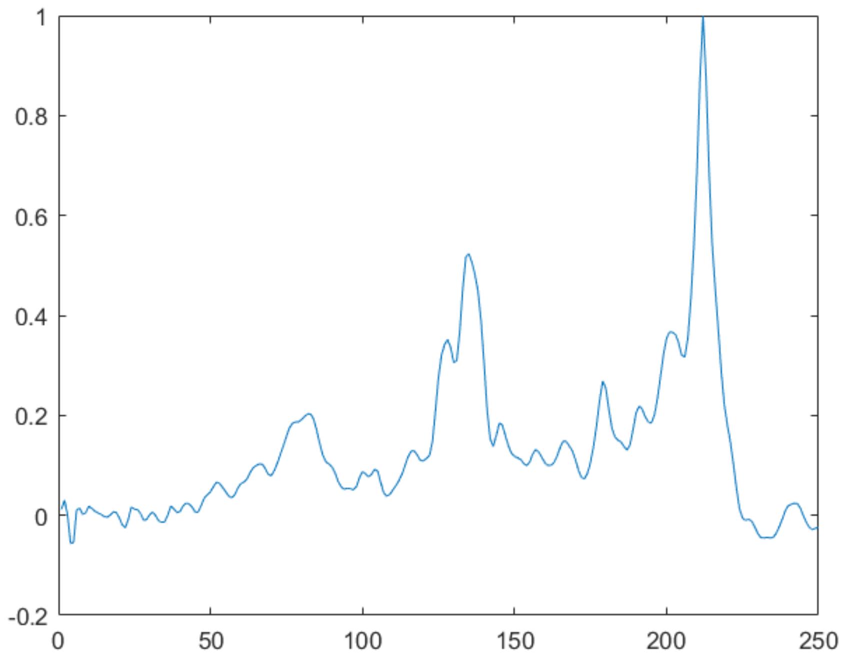


Figure 16: Acoustic signal

After taken the signal as input of the algorithm we got:

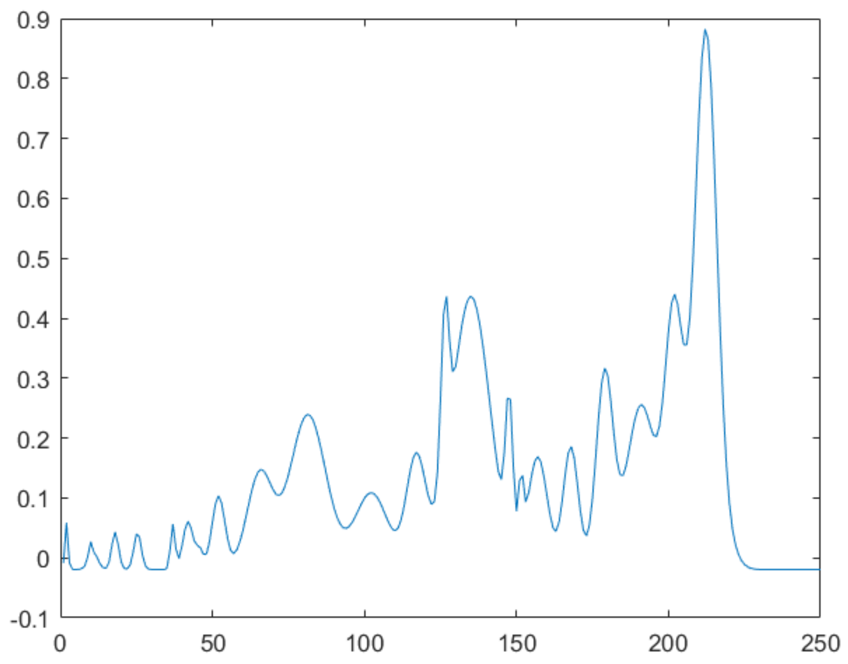


Figure 17: Signal fitting

$$f(x) = -0.019391 + 0.077991 * e^{-\frac{x-8}{2*4.0308^2}} + 0.029313 * e^{-\frac{x-78.015}{2*-12.1833^2}} + 0.074475 * e^{-\frac{x-68.644}{2*2.0904^2}} + 0.062723 \text{*} \mathrm{e}^{-\frac{x-136}{2*8.8707^2}} + 0.030304 * e^{-\frac{x-358.3129}{2*-9.0117^2}} + 0.062352 * e^{-\frac{x-195.0757}{2*9.4636^2}} + 0.074891 * e^{-\frac{x-288}{2*5.9329^2}} + 0.080343 * e^{-\frac{x-328}{2*13.4347^2}} + 0.12219 * e^{-\frac{x-408}{2*17.0122^2}} + 0.15846 * e^{-\frac{x-516.7863}{2*33.3571^2}} + 0.25879 * e^{-\frac{x-643.4006}{2*47.8553^2}} + 0.10977 * e^{-\frac{x-1203.739}{2*6.4559^2}} + 0.12766 * e^{-\frac{x-810.32}{2*43.6617^2}} + 0.18487 * e^{-\frac{x-928}{2*25.7639^2}} + 0.2762 * e^{-\frac{x-1003.6482}{2*-10.4994^2}} + 0.45615 * e^{-\frac{x-1072}{2*49.9795^2}} + 0.24036 * e^{-\frac{x-1172.8112}{2*-9.6008^2}} + 0.18763 * e^{-\frac{x-1248}{2*25.8976^2}} + 0.20467 * e^{-\frac{x-1334.9624}{2*19.8723^2}} + 0.32454 * e^{-\frac{x-1424}{2*22.2661^2}} + 0.27469 * e^{-\frac{x-1520}{2*38.2219^2}} + 0.4226 * e^{-\frac{x-1606.8799}{2*24.0239^2}} + 0.90225 * e^{-\frac{x-1690.0612}{2*29.2977^2}} + 0.015947 * e^{-\frac{x-1761.3146}{2*22.4353^2}} + 0.018299 * e^{-\frac{x-2291.0196}{2*3.375^2}}$$

The error is **0.0014** and the number of peaks fitted is 25.

3.6 Error

At the text we remarked importance of the number of peaks. If we fix the number of peaks error raise as we can see in next figure.

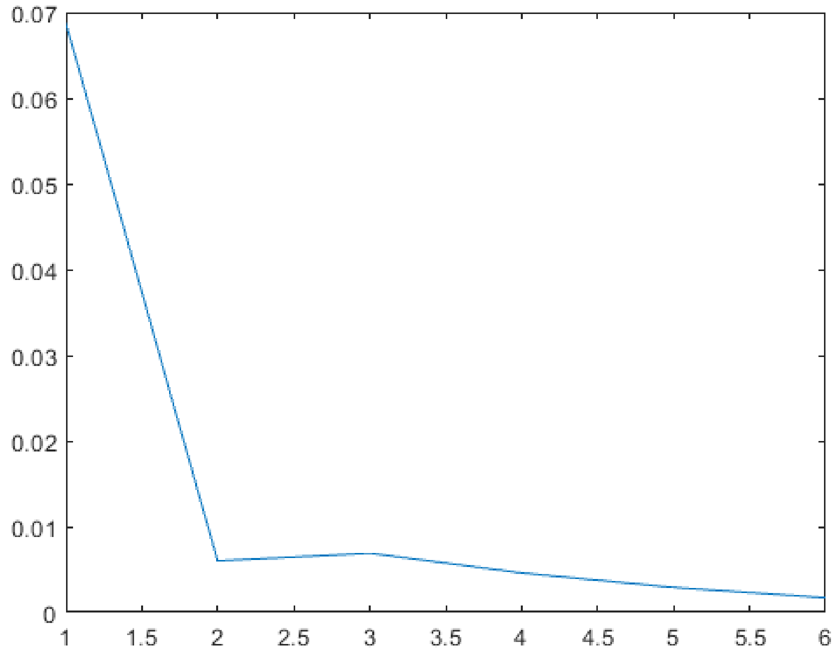


Figure 18: Error by maximum peaks with the 5 peaks signal

In the other hand we show that is useful to fix the number of peaks to avoid the overlapping. So this type of studies, can calculate the minimum number of peaks that the algorithm needs to work. Using the real signal we can get this error curve shown in **(Fig.19)**

This curve shows us that the error stabilized at 20 peaks. Remember that the execution without limitation of this signal gave us 25 peaks, and error of **0.0014**. With the new configuration, limiting the number of peaks to 20, we get from the optimization a lower number of peaks, only 14 and with exactly the same error. Of course this is not going to happen all the time. It uses to give us more error because fixing the maximum number of peaks we restrict the degrees of freedom of the algorithm . The signal fitted with maximum 20 peaks can be seen in **(Fig.20)**

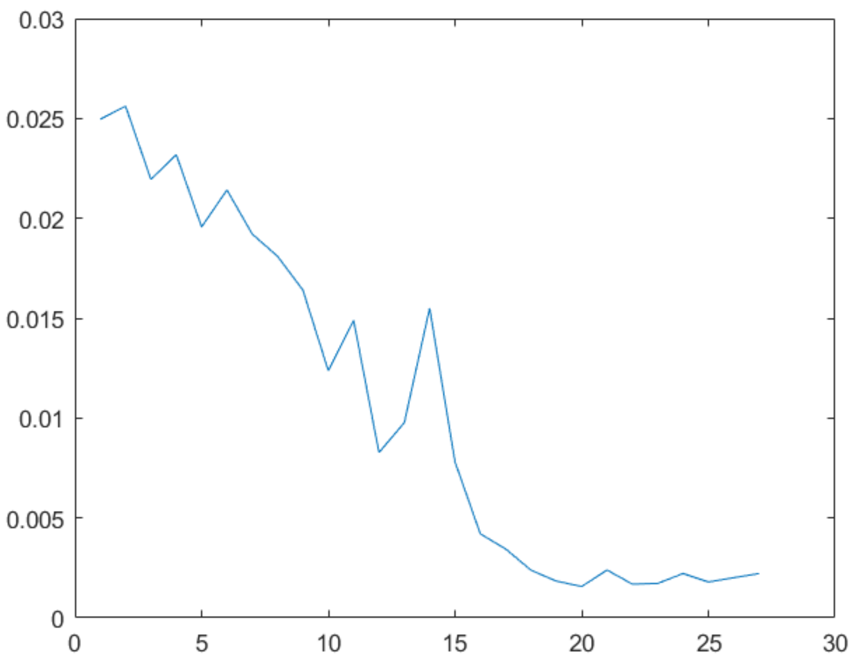


Figure 19: Error by maximum peaks with the real signal

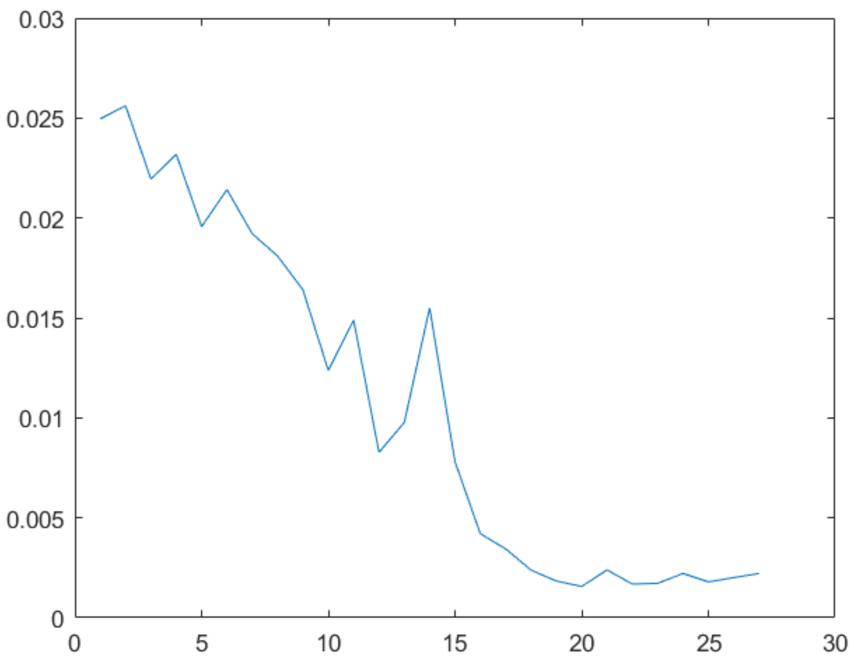


Figure 20: Real signal fit with maximum of 20 peaks

3.7 Population

During all the tests, we used dynamic population size. But we notice that a change in the population affects a lot in the final result, because as K.Messa and M.Lybannon point of in [7], with a small population the convergence is going to be slow, and our stop conditions are limit time or number of generations with a minimum improvement. So the population has a number of individuals with which the error would be stabilized. So we have made the next script, for test it. We are going to use the 5 peaks signals for this test.

Population	Error
1	0.0153
11	0.0077
21	0.0021
31	0.0031
41	0.0014
51	0.0012
61	0.0011
71	0.0012
81	0.0011
91	0.0018

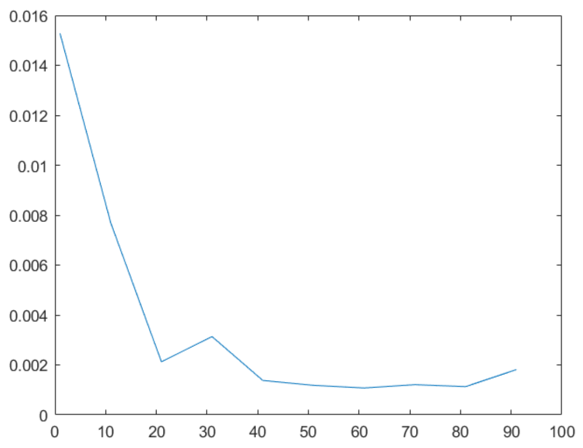


Figure 21: Error by population

3.8 Stability

In genetic algorithm the stability is a weakness comparing to another techniques. For analyzing the stability, first is needed to fix some problems we can have. First one is the overlapping peaks. It can be seen in the double-peak analysis we did above. Second one is the small peaks, that show us a plot with 25 peaks in the real signal (see **Fig.14**), when there are only 20 peaks visible. Another is the extra-range peaks. Both problems can be resolved relaunching the algorithm with the generated signal as input. The error can raise in the new evaluation, but the solution represents graphically the same signal.

4 Conclusion

In this Master's Thesis we designed and developed a method for automatically fitting a acoustic signal, or any type, getting the parameters of the peaks existing in the signal. On the first hand we analyzed the problem, finding the type of model we will use. We choose to make an approximation using the composition of different functions that represent peaks. With this model the system becomes in a non-linear problem. So, we select genetic algorithm techniques for our purpose. First of all we model the system in the way to make it feasible to operate with genetic algorithms. Matlab was chosen as a tool to develop the solution because is the standard programming in most engineering areas. The algorithm was executed in different situations with both artificial signals and real data, in this case coming from acoustics, getting small error values and all the information of the representative peaks of the datasets. Also, it has been found some problems in the stability and in the representation of the solution given by the algorithm. We have supplied a fix for them. In summary, the objective of the work has been achieved designing a set of routines bases in Genetic Algorithms that provide an automatic fitting for peaks in a spectrum.

5 Scripts and functions in Matlab

5.1 Genetic algorithm

```
%INPUT
%S: entrada en formato dos columnas X,Y.
%fixedPeaks: fijar picos m ximos, si pones 0 no se activa
%population: n mero de poblacin  si 0, se calcula dinmicamente
%OUTPUT
%solGen: la solucin  en el formato especficado en la memoria
%numberOfPeaks: el n mero de picos no anulados en la solucin
%error: el error de la solucin
%str: el string de la funcin  resultante listo para latex
function [solGen,numberOfPeaks,error,str] = geneticalFitting(S, fixedpeaks,population)
    inputX = S(:,1);
    inputY = S(:,2);
    %Normalizamos la entrada a valores entre [-1 1].
    norm = max(abs(inputY));
    inputY = inputY/norm;
    Y = inputY;
    X = inputX;
    %El gen es un array que se agrupa cada 4 elementos.
    %El primer elemento del array es el stand y va s lo.
    %Los demas son un valor para saber si existe el pico y la funcin
    %que usa y a, mu y sigma de cada gausiana (o otra)
    %Usamos una funcin  para aportar un primer gen, que se acerque a una
    %solucin.  (para mejorar el rendimiento)
    [defaultGen,numberOfVariables] = calculeDefaultGen(inputX,inputY,fixedpeaks);
    %El n mero de variables es 4 m s 1 para el stand.
    numberOfVariables =(numberOfVariables*4)+1;
    options = gaoptimset(@ga);
    %El tamao  de la poblacin  m nimo es 50 y el m ximo 300. El valor
    %depende del numero de variables siendo multiplicado por 4. Si se
    %tuviera un pc m s potente se puede subir el tamao  de la poblacin.

    if population~=0
        options.PopulationSize = population;
    else
        options.PopulationSize = min([max([50,4*numberOfVariables]),300]);
    end
    %options.Display = 'iter';
    %options.Display = 'none';
    %Parallel Computing Toolbox necesario
    options.UseParallel = 1;
    %Las generaciones sin conseguir mejora antes de para el algoritmo. He
    %pues un valor mayor para problemas pequeos, aprovechando que a m s
    %pequeo  m s  r pido se ejecuta.
    options.StallGenerations = max([10,40-floor(numberOfVariables/10)]);
    %El cambio necesario para contar c mo  mejora, es cercano a 0. Pero
    %mejora poner un valor para que no se estanque con mejoras diminutas.
    options.TolFun = 0.000001;
    %Tiempo l mite del algoritmo para parar. Con los ejemplos aportados no
    %se llega.
    options.TimeLimit = 120;
    options.EliteCount = floor(options.PopulationSize * 0.3);

    %La poblacin  inicial consiste en el gen calculado repetido, y
    %modificado aleatoriamente.
    initialPopulation =repelem(defaultGen,options.PopulationSize,1);
    % Multiplicamos todos los valores de la poblacin  inicial por valores
    % aleatorios E [-2,2];
    aleat = [ones(1,numberOfVariables);2-(rand(options.PopulationSize-1,numberOfVariables)
    options.InitialPopulation = initialPopulation.*aleat;
    tic
    [solGen,error] = ga(@(gen)fitting(X,Y,gen),numberOfVariables,[],[],[],[],[],[],[],optio
    toc
    [str,numberOfPeaks] = writef(solGen);
end
```

5.2 Calculate default gen

```
%INPUT
%X: la frecuencia de la seal
%Y: los valores de la seal
%fixedPeaks: nmero mximo de picos, 0 no se usa
%OUTPUT
%gen: el gen resultante con el que generamos la primera generacin
%numberOfPeaks: nmero de picos que tiene el gen
function [gen,numberOfPeaks] = calculeDefaultGen(X,Y,fixedPeaks)
    %Calculamos el stand con el valor mnimo, y lo aplicamos a toda la
    %onda.
    stand = min(Y);
    Y = Y-stand;
    gen = [stand];
    %Usamos la funcin  nativa findpeaks de matlab. Nos devuelve todos los
    %picos con la altura, la posicin  y una anchura supuesta (este valor no
    %nos sirve).
    [peaks, locs, width, proms]= findpeaks(Y,X);
    numberOfPeaks = length(locs);
    %Si el nmero de picos est fijo, cogemos los que tienen ms valor.
    if fixedPeaks~=0
        [out,indexMax] = sort(peaks);
        indexMax = indexMax(1:min(numberOfPeaks,fixedPeaks));
        peaks = peaks(indexMax);
        locs = locs(indexMax);
        width = width(indexMax);
        proms = proms(indexMax);
        numberOfPeaks = fixedPeaks;
    end
    %Transformamos la informacin  que tenemos en el formato del gen.
    for i = 1:numberOfPeaks
            gen = [gen,[0.01,peaks(i),locs(i),width(i)*0.5]];
    end
```

5.3 Fitting function

```
%INPUT
%X: frecuencia de la seal  original
%Y: valores seal  original
%gen: el gen a evalar
function error = fitting(X,Y,gen)
    %f es la funcin  que transforma el gen en seal.
    yGen = arrayfun(@(x) f(x,gen),X);
    %se calcula el error cuadrtico medio
    error = sum((Y-yGen).^2)/length(X);
end


%INPUT
%x: valor de x a valorar en ese instante
%g: gen a evaluar
%OUTPUT
%y: el valor de x para el gen actual
function y = f(x,g)
    %Separamos el stand (pedestal)
    y = g(1);
    %Seleccionamos segn el valor de la cifra de control
    for i=2:4:size(g,2)
        if g(i)>=0
            if g(i) <= 0.5
                if g(i) < 0.25
%                 Gausiana
                y = y + (g(i+1)*exp((-((x-g(i+2))^2/(2*g(i+3)^2)))));
                else
%                 Laplace
                    y = y +  (1/(2*g(i+1)))*exp(-abs(g(i+2)-x)/g(i+1));
                end
            else
%                Cauchy
                 y = y + ((1/pi)*(g(i+1)/((x-g(i+2))^2)+g(i+1)^2));
            end
        end
    end
end
```

5.4 Stability study

```
%INPUT
%S: entrada en formato dos columnas X,Y.
%fixedPeaks: fijar picos m ximos, si pones 0 no se activa
%OUTPUT
%data: Matriz de 10 columnas con los datos mostrados en los estudios de
%estabilidad de la memoria.
%C mo  plot muestras las muestras que elijas en dos columnas.
function [data] = stabilityStudy(S,fixedpeaks)
    muestras = 10;
    [defaultGen,numberOfPeaksDefault] = calculeDefaultGen(S(:,1),S(:,2),fixedpeaks);
    data = zeros(muestras, numberOfPeaksDefault+3);
    for i =1:muestras
        [gen,data(i,2),data(i,1)] = geneticalFitting(S,fixedpeaks);
        data(i,3) = gen(1);
        for j=2:4:length(gen)
            if gen(j)>=0
                data(i,j+2:j+5) = gen(j:j+3);
            end
        end
        subplot(muestras/2,2,i);
        plot(arrayfun(@(x) f(x,gen),S(:,1)));
        title(strcat('Error: ',num2str(data(i,1)),' Peaks: ',num2str(data(i,2))));
    end
end
```

5.5 Study of error by peaks

```
%INPUT
%S: entrada en formato dos columnas X,Y.
%fixedPeaks: fijar picos m ximos, si pones 0 no se activa
%OUTPUT
%error: Array de longitud la cantidad picos de S, cada celda significa un
%pico m s. Contiene la media de las muestras para esa configuracin  de
%picos.
function [error] = errorByPeaks(S)
    muestras = 4;
    [defaultGen,numberOfPeaksDefault] = calculeDefaultGen(S(:,1),S(:,2),0);
    error = zeros(numberOfPeaksDefault,1);
    for numberOfPeaks =1:numberOfPeaksDefault
        data = zeros(muestras, numberOfPeaksDefault);
        for i =1:muestras
            [gen,data(i,2),data(i,1)] = geneticalFitting(S,numberOfPeaks,0);
            data(i,3) = gen(1);
            for j=3:4:length(gen)
                if gen(j-1)>=0
                    data(i,j:j+3) = gen(j-1:j+2);
                end
            end
        end
        mediError = mean(data(:,1));
        error(numberOfPeaks) =mediError(1);
    end
end
```

5.6 Study of error by population

```
%INPUT
%S: entrada en formato dos columnas X,Y.
%max: fijar picos m ximos, si pones 0 no se activa
%step: el aumento de poblacin  en cada ciclo
%OUTPUT
%error: Array de longitud max, cada celda significa un
%step m s por poblacin.  Contiene la media de las muestras para esa configuracin  de
%poblacin.
function [error] = errorByPopulation(S,step,max)
    muestras = 4;
    [defaultGen,numberOfPeaksDefault] = calculeDefaultGen(S(:,1),S(:,2),0);
    error = zeros(numberOfPeaksDefault,1);
    for numberOfPop =1:step:max
        data = zeros(muestras, numberOfPeaksDefault);
        for i =1:muestras
            [gen,data(i,2),data(i,1)] = geneticalFitting(S,0,numberOfPop);
            data(i,3) = gen(1);
            for j=3:4:length(gen)
                if gen(j-1)>=0
                    data(i,j:j+3) = gen(j-1:j+2);
                end
            end
        end
        mediError = mean(data(:,1));
        error(numberOfPop) =mediError(1);
    end
end
```

6 References

[**1**] Levy, E. C. (1959). Complex-curve fitting. IRE transactions on automatic control, (1).

[**2**] G. Trejo-Caballero, H. Rostro-Gonzalez, C. H. Garcia-Capulin, O. G. Ibarra-Manzano, J. G. Avina-Cervantes, and C. Torres-Huitzil (2015). Automatic Curve Fitting Based on Radial Basis Functions and a Hierarchical Genetic Algorithm. 2015 Hindawi Publishing Corporation

[**3**] Colin Hansen, Scott Snyder, Xiaojun Qiu, Laura Brooks, Danielle Moreau. (2012) Active Control of Noise and Vibration. Volume I, CRC Press. USA

[**4**] Guo, Hongwei. (2011). A Simple Algorithm for Fitting a Gaussian Function [DSP Tips and Tricks]. IEEE Signal Processing Magazine - IEEE SIGNAL PROCESS MAG. 28. 134-137. 10.1109/MSP.2011.941846.

[**5**] Alvarado-Durán, Pablo A., Álvarez-López, Mauricio A., Orozco-Gutiérrez, Álvaro A. (2013). Detección de Eventos Sonoros en Señales de Música Usando Procesos Gaussianos. Tecno Lógicas, (31), 93-122. Retrieved August 22, 2018, from
http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-77992013000200006
&lng=en&tlng=es.

[**6**] Deb, K. Sadhana (1999) An introduction to genetic algorithms.
https://doi.org/10.1007/BF02823145

[**7**] M.Gulsen, A.E.Smith and D.M.Tate (1995) A genetic algorithm approach to curve fitting. INT.J.PROD.RES, VOL.33, No.7, 1911-1923

[**8**] K.Messa, M.Lybanon (1992) Curve fitting using genetic algorithms. Naval Oceanographic and Atnospheric Research Laboratory, Stennis Space Center.

[**9**] Felix Scholkmann, Jens Boss and Martin Wolf (2012) An Efficient Algorithm for Automatic Peak Detection in Noisy Periodic and Quasi-Periodic Signals. Biomedical Optics Research Laboratory, Division of Neonatology, University Hospital Zurich.

[**10**] C. L Karr, D. A. Stanley, and B. J. Scheiner (1991) Genetic Algorithm Applied to Least Squares Curve Fitting. U.S.Bureau of Mines. Spokane Research Center.