Part I

Localization

This part of the project takes care of the most important aspect of Acoustic Emission inspection, i.e the Localization of the AE event. There are several techniques to handle such tasks; however, we are going to limit ourselves to most basic one,Time Difference of arrival. TDoA alone will not be much more beneficial as some noise and reading error are involved in the processes. As a result, an optimization algorithm has to arise to solve such a problem (all about it in optimization part). As all the materials have not the same inside-structure nor all directions within the same material are the same, we have to put some assumptions to help us unify our model.

0.0.1 Assumptions

- Wave path is constant; otherwise, the solution is not going to be the same for a given set of data.
- Wave velocity is the same everywhere on structure.
- Structure thickness is uniform. If not, a third dimension would be involved, which only complicates the model
- Material is homogeneous.

0.0.2 Techniques

There are plenty of methods to locate the source of an acoustic emission. The most basic ones are Time of arrival (ToA), do not confuse with arrival time which is signal parameter, and Time difference of arrival (TDoA).

Time of Arrival (ToA)

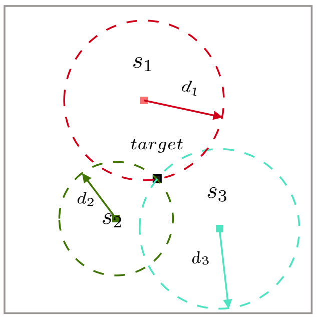
This is a fairly easy technique to implement and derive a general equation that describes this simple behavior. ToA is based on knowing the exact time the signal has been sent and the exact time the signal has reached a reference point in space, which could be a sensor. Also, knowing the velocity of signal through the material, we can compute the the distance between the event position and the reference point through the following equation:

$$d = velocity.(t_{arrival} \ - \ t_{sent})$$

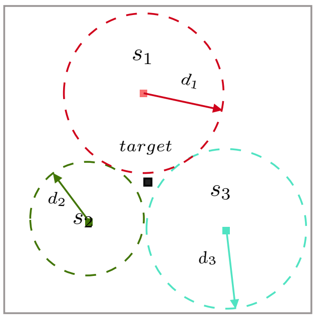
Nevertheless, the equation above can be simple in the 1-D case, it yields an equation for circle in two dimensions.

$$d_i = \sqrt{(x_i \; - x_s)^2 + (y_i \; - y_s)^2}$$

where, (x_i, y_i) is the location of the i^{th} sensor on the plane and (x_s, y_s) is the AE source unknown location.Once this equation is calculated for enough reference points (sensor locations) on the same plane, the location of the AE event cab be easily obtained from the intersection point of the circles. It has to be noted that at least 3 sensors are required for the two dimensions to kill the symmetry around the line joining the original two sensors, in the same way 4 sensors for 3-D. Despite its simplicity, ToA can have some accuracy-relates issues. The first sub-figure represents the ideal case, where all the circles intersect in one and only one point (the target), but this rarely happens, the second sub-figure, on the other hand, shows the case where the circle do not meet in any point; however, there is a margin of error we can search in. This problem arises due to some issues in time synchronization and acquisition errors.



(a) Ideal case of ToA solution



(b) non-ideal case of ToA

Figure 1: ToA solution visuliazation

Time Difference of Arrival (TDoA)

TDoA is the second most popular technique, it is somehow more versatile than ToA as this method does not require the time that the signal was sent from the AE source, only the arrival time and signal speed are needed to localize the source. Once two sensors at least (in 1-D) receives the signal, we can compute the difference in arrival time and hence, the difference of distance between the sensors and target. This can follow the upcoming equation:

$$\Delta d_{i,j} = v \; . \; \Delta t_{i,j}$$

In the 2-D case, however, the equation can be written as follows:

$$\Delta d_{i,j} = \sqrt{(x_i \; - x_s)^2 + (y_i \; - y_s)^2} - \sqrt{(x_j \; - x_s)^2 + (y_j \; - y_s)^2}$$

$$where: \quad 1 \leq i, j \leq n \quad , \; i \neq j$$

And n is the number of sensors in the sensor array. DToA is quite hard to solve as there is not an obvious closed form solution . The solution can rely on an optimization algorithm, which , in turn, tries to minimize the difference between the calculated time difference and the observed one:

$$f(x, y) = \sum_{i=1}^{n} (\Delta t_{i,obs} - \Delta t_{i,clc})^2$$

Where:

$$\Delta t_{i,obs} = t_i - t_1$$

$$\Delta t_{i,clc} = \left[\sqrt{(x_i \; - x_s)^2 + (y_i \; - y_s)^2} - \sqrt{(x_1 \; - x_s)^2 + (y_1 \; - y_s)^2} \; \right] / v$$

So our main goal is :

$$\min_{x,y \; \in \; R^2} \{f(x, y)\}$$

And this will be discussed in the optimization part. Now, the important question is how to compute t_i for a given waveform. Some methods are going to be introduced in the next section.

0.0.3 Computation of arrival time

Threshold Crossing

Generally, there are two main techniques used to compute the arrival time of the signal to a given sensor. The first one is threshold crossing, where a fraction α of the maximum amplitude is set as a minimum threshold, and then we have to iterate over the data points until we have the first crossing. It is somehow a simple approach, and not computationally expensive. In fact its complexity is fraction of n or formally $\theta(n)$, where n is the number of samples in the waveform.

$\rightarrow$ threshold figure here

The downside of this method is accuracy as it might skip some data points , hence an increase in arrival time. That is because it is very sensitive to the parameter α , and here comes the second approach (AIC).

Akaike Information Criterion (AIC)

The Akaike Information Criterion (AIC) function was first originated in information theory and used as an estimator for relative quality of different statistical models. It can be useful to compute the arrival time of the signal to the sensor of interest through the following equation:

$$AIC(t) = t.\log_{10}\left(var(x[1:t])\right) + (T-t-1).\log_{10}\left(var(x[t:T])\right)$$

where T is the signal duration, and var is the classical statistical variance. Since the function splits the signal into two-time vectors $[1 : t]$ and $[t : T]$ and describes the similarity between them. When point t is aligned with the signal onset, the vector prior to t contains only high-entropy uncorrelated noise and the vector after t contains only low-entropy signal with marked correlation and the AIC function therefore returns a minimum. on the contrary, AIC is computationally expensive as it splits the waveform into two vectors for every time point, in fact, its algorithmic complexity can be formulated as $\theta(n^2)$. This can affect the runtime of the program especially if it is utilized in real-time; however, there is a little trick, i.e. caching the AIC data in (for example) a JSON file and loading the data after computing the function for just one time per sensor data. AIC proved useful with reasonable accuracy measures when it is compared to the threshold crossing. To sum up, there is a trade off between accuracy and computational time in two methods.

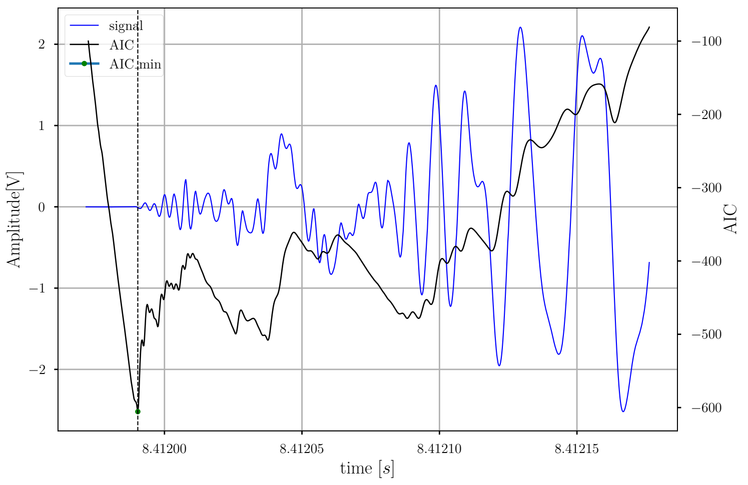


Figure 2: AIC implementation