Mathematica scripts for searching wave features in a wave train sample

Script features:

-Searches for a couple of Maximums and Minimums in a wave sample time.
-After identifying the Maximums and Minimums it calculates the wavelength and velocity of the waves passing thought the wave gauges.
-It will create an array to store the Maximums, Wavelengths and velocities in a .m file.

What is need it:

-Each wave trains needs to be in a CSV file, with x,y data on each column.
-Wave train boundaries need to be specified by the user changing the boundary variables, ci1, bi1, ai1, cs1, bs1, cs1, ci2, bi2, ai2, cs2, bs2, as2. Basically each one is just the inferior and superior boundaries of our wave, our wave is divided in two zones, one from the 1st trough to the crest and the second one from the crest to the 2nd trough.
-Each wave probe data has to be in its own cvs file.
-X data (time is on 1/100 seconds, meanwhile Y data (height) is on millimetres.

What can be done:

-The wok can be replicated on python, C or any other programming language.
-The scripts can add a noise filter [this is added in the work for the 2nd set of experiments]. The noise filter needs to be a low band filter due to the small fluctuation cause by the cases at the wave gauge being on several scales much more lower than the wave signal, usually any less than a period of .1 seconds will suffice.
-Scripts can be automatise to search for waves before the reflection time if this one is known beforehand.
-The wave train a analysis can also automatise with need to do it by hand using a singles script with data from a .cvs file.
-Signal could be decompose easily with filters to obtain different waves moving on the same train, this will give the composition of an irregular one using linear functions. This is after a trigger on data major than certain mean value tell us there is a wave train present on the data line.

The Code is simple and requires some handwork cause each wave train boundary needs to be defined by hand, this was done cause the reflections and the properties of each wave train where sometimes irregular and a wave had to be choose. Waves where choose after carefully disregarding waves that had reflection components or that where not fully developed due to the wave tank inner workings.

Code sample:

1st block:

During the 1st block, the main variables are set and data is imported from cvs files in order to be analysed. Also the boundaries of X (time) where our wave lives are set.

```
SetDirectory[“/Users/sialuk/Desktop/laboratory1sttest/mathematica2/"]
```

“Sets the directory path where mathematica will search for the files”

```
Cdatam = Import["Test1sortedC.csv", {"Data", Range[10, 1000]}];
Bdatam = Import["Test1sortedB.csv", {"Data", Range[10, 1000]}];
Adatam = Import["Test1sortedA.csv", {"Data", Range[10, 1000]}];
Cmean = Cdatam[[All, 2]];
Bmean = Bdatam[[All, 2]];
Amean = Adatam[[All, 2]];
```

%datam [Cdatam, Bdatam, Adatam]: are arrays to store data from the first seconds of testing when the water is still, this data will be used to calculate a mean value. The wave mean value of the 1st seconds will be used to plot the data correctly, see fig 1.

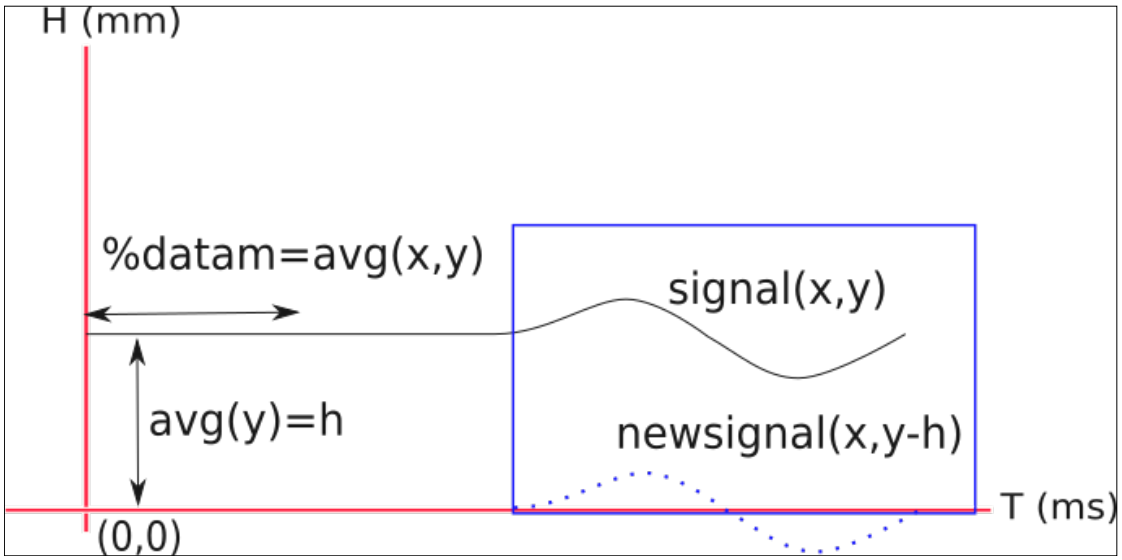

Fig1.- Due to the difference on each wave gauge signal, every signal is moved h units from the 0 level. In order to move the signal to 0 a mean is calculated the first 10 seconds of each experimental unit, the mean in Y then will be subtracted form the height data to move our signal to level 0. Same procedure will be done with each wave gauge.

%mean : [Cmean, Bmean, Amean]: The arrays will sure only the Y data to be averaged and then used to plot correctly the signal.

Due to the high degree of harmonics the signal had to be most of the time analysed one wave train each time, a value on time in milliseconds was chosen for the inferior and superior boundaries on each wave train.

```
cs1 = 2500;
bs1 = 2600;
as1 = 2700;

ci1 = 2400;
bi1 = 2500;
ai1 = 2600;
```

%s1-%i1: [cs1, bs1, as1, ci1,bi1, ai1]: These variables are the superior (s) and inferior (i) time for each analysis on the experiment number (1), see fig 2.

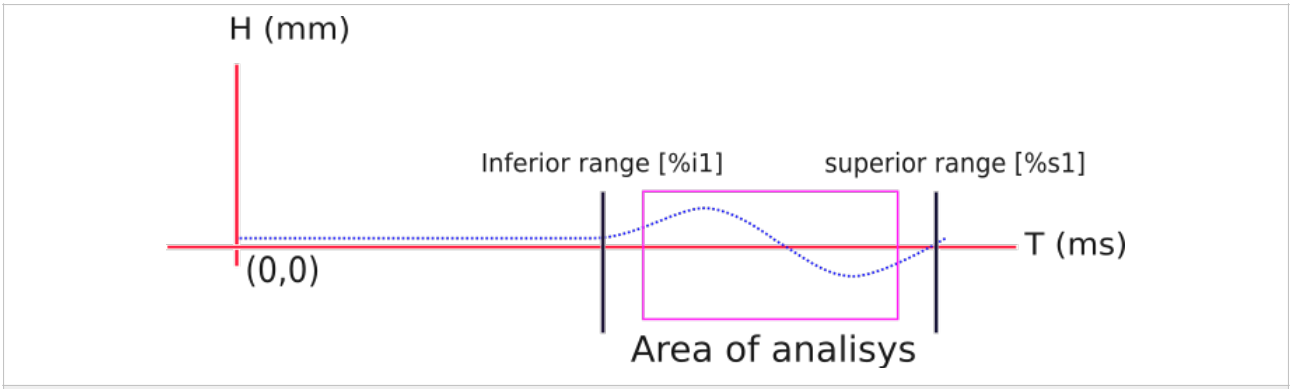

Fig 2.- Each wave train has a defined area choose to be analysed individually. The area is marked at the script by a variable defining an inferior range and a superior range on X (time variable).

```
datactest1 = Import["Test1sortedC.csv", {"Data", Range[ci1, cs1]}];
datcx = datactest1[[All, 1]];
datcy = datactest1[[All, 2]];
cymean = Mean[Cmean];
datcy = datcy - Abs[cymean];
datactest1 = Transpose[{datcx, datcy}];
Length[datactest1]*100;

databtest1 = Import["Test1sortedB.csv", {"Data", Range[bi1, bs1]}];
datbx = databtest1[[All, 1]];
datby = databtest1[[All, 2]];
bymean = Mean[Bmean];
datby = datby + Abs[bymean];
databtest1 = Transpose[{datbx, datby}];
Length[databtest1]*100;

dataatest1 = Import["Test1sortedA.csv", {"Data", Range[ai1, as1]}];
datax = dataatest1[[All, 1]];
datay = dataatest1[[All, 2]];
aymean = Mean[Amean];
datay = datay + Abs[aymean];
dataatest1 = Transpose[{datax, datay}];
Length[dataatest1]*100;
```

data%test1: [dataatest1,databtest1,datactest1]: Are the variables that store the arrays containing the data from wave probes A,B,C. Each array goes from the inferior range to superior one. The data is imported from a .csv file.
dat%x-dat%y: [datay, datby, datcy, datax, datbx, datcx]: The array is divided in their sections x and y to work on each one individually.

%mean: [amean, bmean, cmean]: each sample from the first seconds on each wave prove is averaged with function Mean [].

dat%y: the data for the variables on y for each wave probe is then corrected after displacing it with the mean value of each wave probe first seconds, see figure 1.

data%test1: The data is again joined transposing the two arrays and then is measured to see if has the correct number of members with the command Lenght[].

```
{minci} = {MinimalBy[#, Last]} &@datactest1;
{minbi} = {MinimalBy[#, Last]} &@databtest1;
{minai} = {MinimalBy[#, Last]} &@dataatest1;


Cplot = ListLinePlot[datactest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@minci}]
Bplot = ListLinePlot[databtest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@minbi}]
Aplot = ListLinePlot[dataatest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@minai}]


Mindatactest1yi = minci[[All, 2]];
Mindatabtest1yi = minbi[[All, 2]];
Mindataatest1yi = minai[[All, 2]];
avgMindatactest1yi = N[Mean[Mindatactest1yi]];
avgMindatabtest1yi = N[Mean[Mindatabtest1yi]];
avgMindataatest1yi = N[Mean[Mindataatest1yi]];

Mindatactest1xi = minci[[All, 1]];
Mindatabtest1xi = minbi[[All, 1]];
Mindataatest1xi = minai[[All, 1]];
avgMindatactest1xi = N[Mean[Mindatactest1xi]];
avgMindatabtest1xi = N[Mean[Mindatabtest1xi]];
avgMindataatest1xi = N[Mean[Mindataatest1xi]];
```

min%: [minc,minb,mina]: are the values for the 1st maximum on the specific wave train time, this for each data set data%test1.

%plot: [Cplot, Bplot, Aplot]: A plot is made from all data sets and a marker is put to verify each maximum on the wave, see fig 3. From here the array name needs to be used data%test1 and the plot range on {x, y} has to be specified. The command ListLinePlot needs to take two arguments, the 1st one is the name of the array to be plotted, in this case the variable containing the datasets, and second the range to plot on {{x inferior, x superior} , {y inferior, y superior}}. See figure 3.

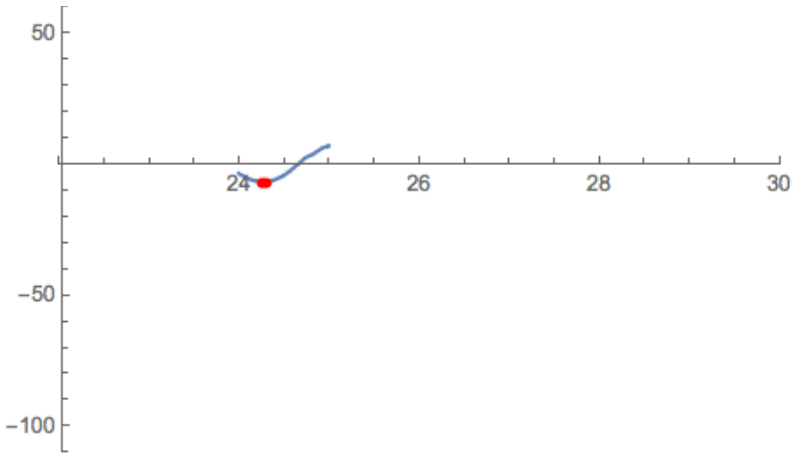

Fig 3.- The 1st minimum is found, this minimum will be used to calculate variables like the wavelength, wave velocity and wave size after comparing it with the local maximum (crest) and the 2nd local minimum.

2nd block

On the 2nd block the analysis moves to the second part of our wave, this by dividing the wave in two sections with the crest as the middle part.

```
cs2 = 2600;
bs2 = 2700;
as2 = 2800;


ci2 = cs1;
bi2 = bs1;
ai2 = as1;
```

Here the superior boundary is changed and the inferior boundary is now the inferior boundary of our wave zone, see figure 4.

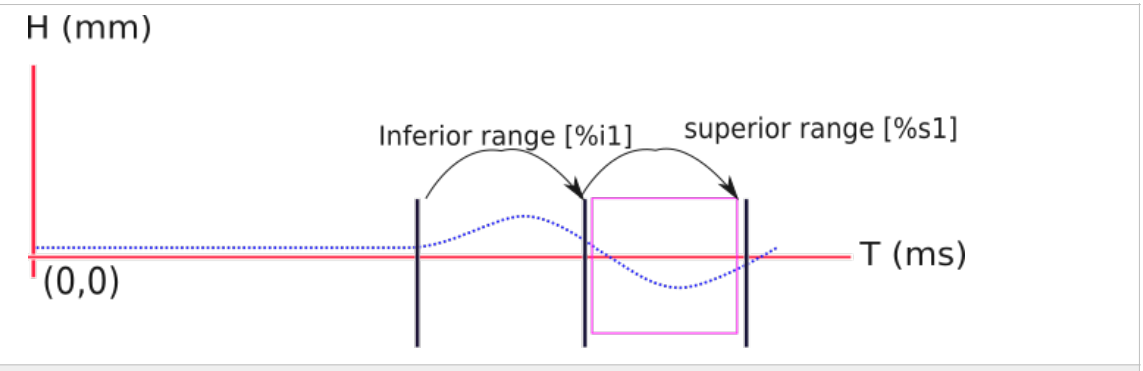

Fig 4.- The boundaries are changed, now to analyse the 2nd part of the wavier search of the second trough velocity and position.

```
{maxc} = {MaximalBy[#, Last]} &@datactest1;
{maxb} = {MaximalBy[#, Last]} &@databtest1;
{maxa} = {MaximalBy[#, Last]} &@dataatest1;
```

max%: [maxc,maxb,maxa]: are the values for the 1st maximum on the specific wave train time, this for each data set data%test1.

```
Cplot = ListLinePlot[datactest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@maxc}]
Bplot = ListLinePlot[databtest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@maxb}]
Aplot = ListLinePlot[dataatest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@maxa}]
```

%plot: [Cplot, Bplot, Aplot]: A plot is made from all data sets and a marker is put to verify each maximum on the wave, see figure 5. From here the array name needs to be used data%test1 and

the plot range on {x, y} has to be specified.

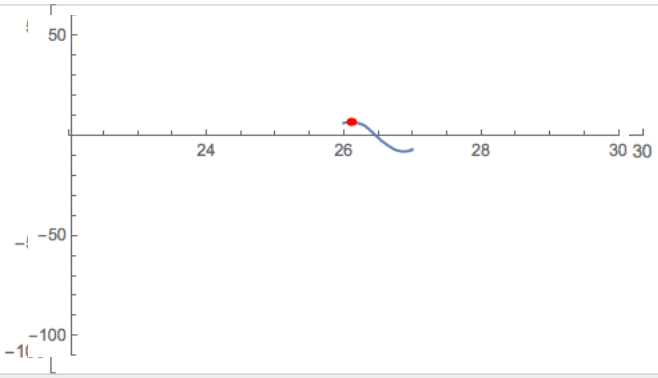

Fig 5.- Plot showing the maximum of the wave on the bounded range by %s1 and %i1.

```
Maxdatactest1y = maxc[[All, 2]];
Maxdatabtest1y = maxb[[All, 2]];
Maxdataatest1y = maxa[[All, 2]];
```

Maxdata%text1y: [Maxdatactest1y, Maxdatactest1y, Maxdatactest1y]: Cause is usual that we have several maximums on our data sets due to sampling at 100 hhz all of them are stored.

```
avgMaxdatactest1y = N[Mean[Maxdatactest1y]];
avgMaxdatabtest1y = N[Mean[Maxdatabtest1y]];
avgMaxdataatest1y = N[Mean[Maxdataatest1y]];
```

avg*Maxdata%text1y: [avgMaxdatactest1y, avgMaxdatactest1y, avgMaxdatactest1y]: Each maximum appears at some time t around the main crest of our wave, this is due to the signal noise, several maximums appear at t+Δτ. This delta is small enough that we can make an average of all points and then get the average time of our crest with a deviation of not more than ms.*

```
Maxdatactest1x = maxc[[All, 1]];
Maxdatabtest1x = maxb[[All, 1]];
Maxdataatest1x = maxa[[All, 1]];
avgMaxdatactest1x = N[Mean[Maxdatactest1x]];
avgMaxdatabtest1x = N[Mean[Maxdatabtest1x]];
avgMaxdataatest1x = N[Mean[Maxdataatest1x]];

velchbc = 2/(avgMaxdatabtest1x - avgMaxdatactest1x);
velchab = 2/(avgMaxdataatest1x - avgMaxdatabtest1x);
```

Maxdata%text1x: [Maxdatactest1x, Maxdatactest1x, Maxdatactest1x]: The time (coordinate x) of our array is tired for each maximum found around the crest main point, then they are averaged again.

velch%%: [velchab, velchbc]: Cuase our wave probes are at a distance of two meters each the averaged time for each wave peak is used to calculate the velocity, see figure 6.

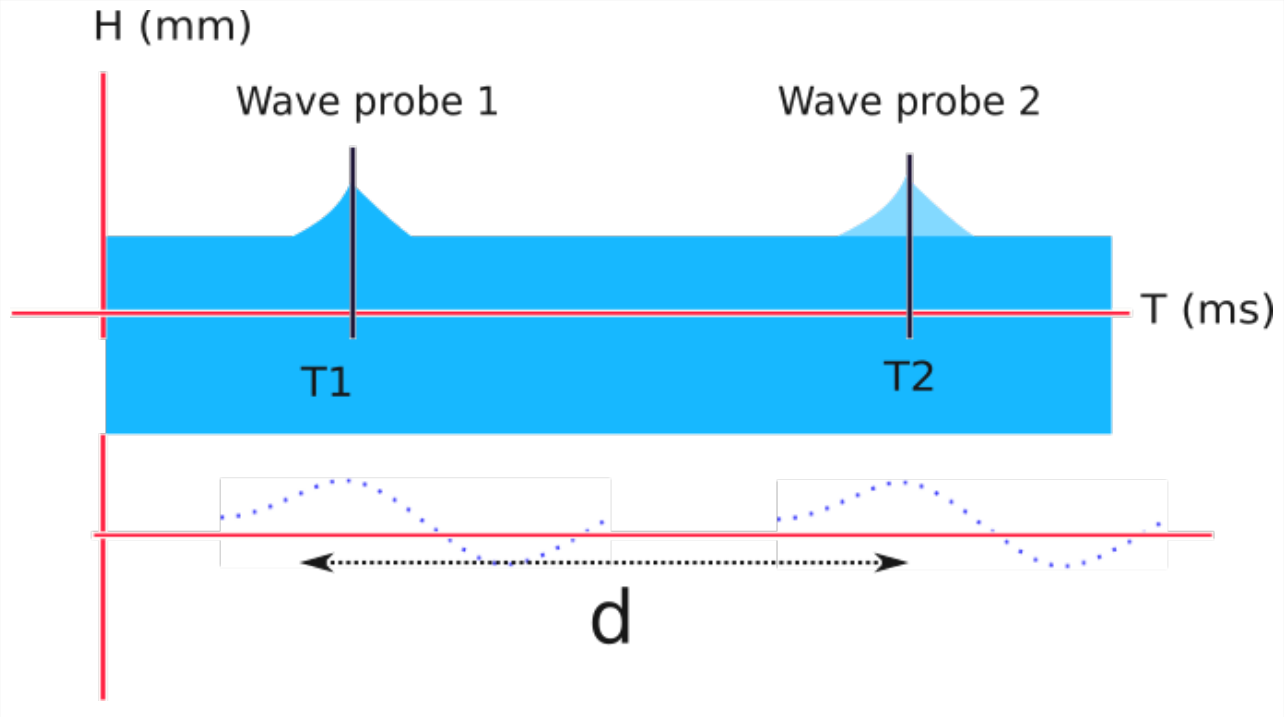

Fig 6.- Cause the wave appears a two different times on each wave probe (coordinate x of our array), then the difference on time between each wave probe is the Δτ. Then as we know each wave probe is at 2 meters from each other 2/Δτ. will give us the velocity of the train after passing from C sensor to B, and also from B sensor to C.

3rd block

Exactly same procedure is applied to search now for a superior minimum, this will be the superior bound of our wave. in this section the wavelengths are calculated along with the creation of the arrays for each variable that will be plotted afterwards.

```
{mincs} = {MinimalBy[#, Last]} &@datactest1;
{minbs} = {MinimalBy[#, Last]} &@databtest1;
{minas} = {MinimalBy[#, Last]} &@dataatest1;


Cplot = ListLinePlot[datactest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@mincs}]
Bplot = ListLinePlot[databtest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@minbs}]
Aplot = ListLinePlot[dataatest1, PlotRange -> {{22, 30}, {-110, 60}},
  PlotLegends -> "Load Coefficient",
  Epilog -> {PointSize -> Medium, Red, Point@minas}]



Mindatactest1ys = mincs[[All, 2]];
Mindatabtest1ys = minbs[[All, 2]];
Mindataatest1ys = minas[[All, 2]];
avgMindatactest1ys = N[Mean[Mindatactest1ys]];
avgMindatabtest1ys = N[Mean[Mindatabtest1ys]];
```

```
deltatimeb = avgMindatabtest1xs - avgMindatabtest1xi;
deltatimea = avgMindataatest1xs - avgMindataatest1xi;


λc = velchbc*deltatimec;

λb = (velchbc + velchab)/2*deltatimeb;

λa = velchab*deltatimea;

Hratioc = avgMindatactest1yi - avgMaxdatactest1y
Hratiob = avgMindatabtest1yi - avgMaxdatabtest1y
Hratioa = avgMindataatest1yi - avgMaxdataatest1y


listλc1 = {λc}
listλb1 = {λb}
listλa1 = {λa}
listvelbcnew1 = {velchbc}
listvelabnew1 = {velchab}
listMaxc1 = {avgMaxdatactest1y}
listMaxb1 = {avgMaxdatabtest1y}
listMaxa1 = {avgMaxdataatest1y}
listHratioc12 = {Hratioc}
listHratiob12 = {Hratiob}
listHratioa12 = {Hratioa}
```

deltatime%: [deltatimec, deltatimeb, deltatimea]: The difference of time when our two trough pass on each sensor is calculated simply subtracting time form each one.

λ%: [λa, λb, λc]: The wavelength is calculated multiplying the velocity of the wave and the difference of time between each trough passing at our sensor. This can be explained at detail in figure 7.

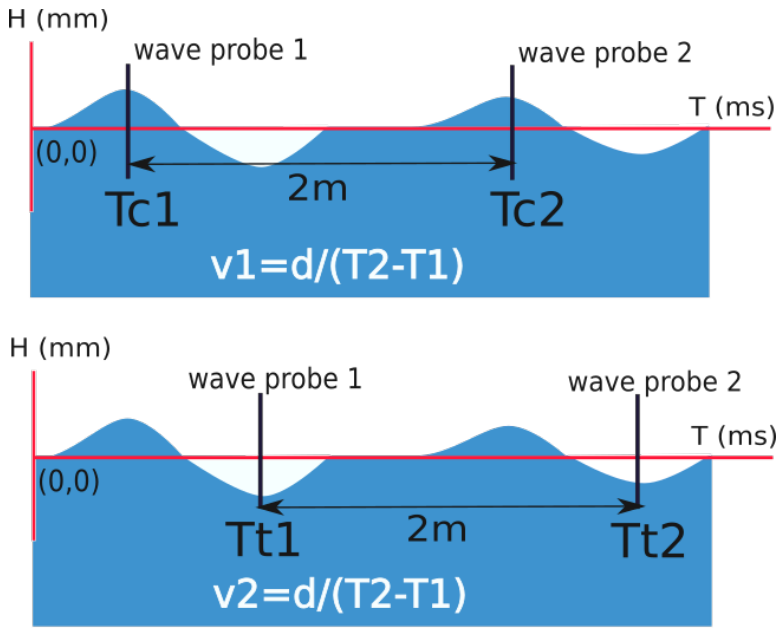

Fig 7.- Apart from the change in velocity if we measure the change on time from crest to trough we ca notice an expansion or contraction of the wave train. Using this change in time when the crest and trough pass at a determined velocity on a determined distance between each sensor we get the wavelength.

Hratio%= [Hratioc, Hratiob, Hratioa]: Apart from the maximums of each wavelength being calculated for that specific wave, the ratio between the maximum and the thought is specified. The reason for this, is because of shoaling the peak of the wave will decay but also the trough will attenuate giving a better value if we take the wave trough-peak ratio than just the maximum. See figure 8.

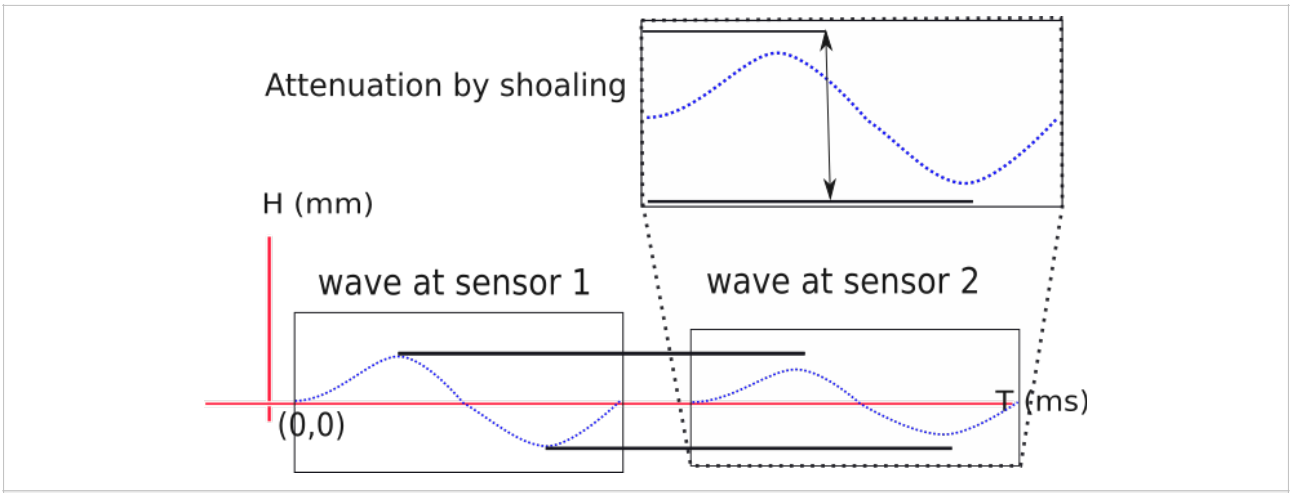

Fig. 8.- The wave crest size sometimes it appears to be bigger at our experimental data than it was at the beginning, however this only applies for the crest, if we then include the attenuation of the trough our data clearly delivers a wave smaller than the one being produced by the wave maker. This is cause the wave crest appears to increase but also the trough attenuates its depth.

list%%%%: lists are created to store the data gathered, like the wavelength, velocity and maximums.

Particularities block

Cause each wave train is analyse separately then at the 1st script the function to create a lost is used:

```
listHratioa12 = {Hratioa}
```

That is:

```
“listname” = {variable to put on the list}
```

Then at the next script the next variable needs to be added at the end of the list using a new function called insert:

```
listHratioa12 = Insert[listHratioa12, Hratioa, -1]
```

That is:

```
“listname”=Insert[“listname”, v”ariable to insert on list”, “position where it will be inserted”]
```

The procedure needs to be repeated each time a new script is called to obtain new variables to the list before export the file as .csv using:

```
Export["listc12ratioa.mx", listHratioa12]
```