A microseismic simulation for algorithm testing

Michael P. Lamoureux, Rituraj Shukla[*], Gary F. Margrave

ABSTRACT

A physical model of a microseismic array is described, used as a simple test platform to design and evaluate algorithms for locating microseismic events. Some initial testing was done, using cross correlation and phase transform methods to measure time delays, and Gauss-Newton minimization to locate the simulated acoustic event.

In summer 2011 as part of a Globalink undergraduate student research project, a DSP system interfaced to a small array of microphones was constructed and tested, to locate acoustic events in air based on measured time delays in recorded signals. A number of array geometries were designed, different delay measuring algorithms implemented, and numerical methods evaluated. The goal was to simulate the computational effort needed in a real microseismic experiment, using a simple physical device that is easily tested in a student laboratory.

INTRODUCTION

Microseismic sensing is an essential technology for monitoring small seismic events that occur within geological formations, and in particular in hydrocarbon reservoirs, to identify and track physical processes that occur over time. Such methods are useful to monitor oil production, CO2 storage, fracking, and other industrial processes. Events are monitored via the recording of seismic data from an array of geophones situated around the geological formation or subsurface reservoir, and applying certain numerical algorithms to the recorded signals in order to locate and characterize the event.

Figure 1 gives a simple schematic of a microseismic array setup. A collection of geophones is placed on the surface of the earth; they could also be down a bore hole. An event occurs under the surface that generates a seismic signal, which the geophones then record. Signal processing algorithms are applied to extract useful information from the signals, such as the location and magnitude of the event.

Since it is an expensive venture to install and monitor a geophone array at a real earth site, for training and development purposes, it is useful to have a simple laboratory model to quickly test some ideas related to extracting information from a microseismic monitoring experiment. The goal in this short project was to create a signal processing system based on a microphone array, to monitor acoustic events in air, and derive location information from the recorded signals using some simple numerical algorithms.

Figure 2 shows the first test bed constructed for the project. A linear array of three electret-style condenser microphones was mounted on a ruler, with electrical leads going to a multi-channel analog-to-digital converter (a National Instruments device). Such a linear

[*]Indian Institute of Technology, Mumbai.

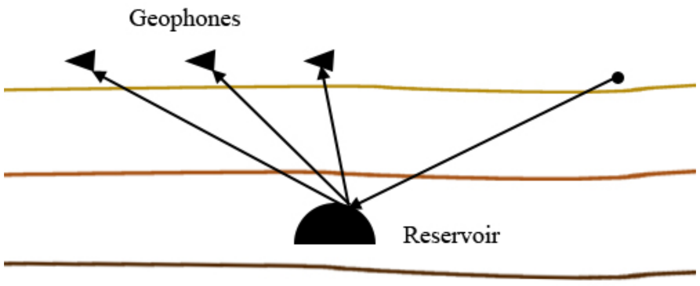


FIG. 1. An array of geophones, to locate an event

array is sufficient to locate acoustic events in a two dimensional plane. Figure 3 shows a 2D array of six microphones, which is a geometry that can be used to locate events in the full three dimensions. Figure 4 shows the six element array connected to the National Instruments multi-channel ADC which is used to digitize and store the signals received by the microphones.

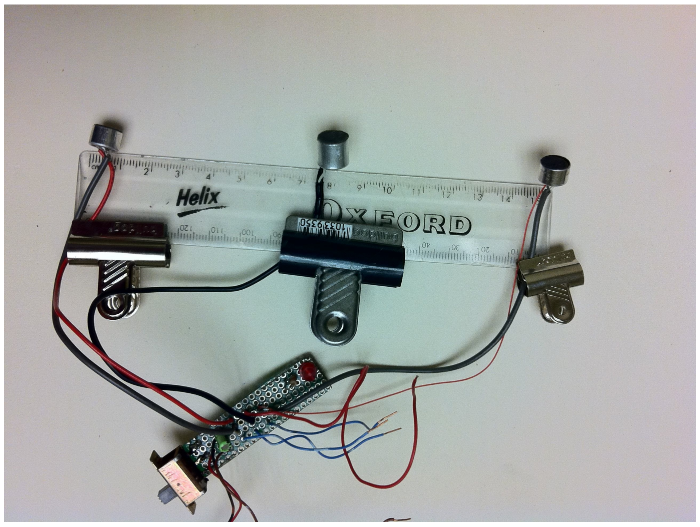


FIG. 2. A linear array of 3 microphones

The project aims at simulating the microseimic monitoring problem in a simple way. The entire setup is done in a room where the speed of sound in air is constant, in contrast to the actual problem where the velocity changes due to the presence of different rock types beneath the earth's surface. Microphones are use to record and approximate the location of the sound source, rather than an array of geophones. The microphones are arranged in a fixed geometry with given positions and spacing, so the sound reaches each of them individually at different times. The relative delay in recorded sound event is used to determine the coordinates of the sound source.

There are several methods to estimate the delay. For this initial project, two methods were implemented to find the relative time delay: cross correlation and phase transform

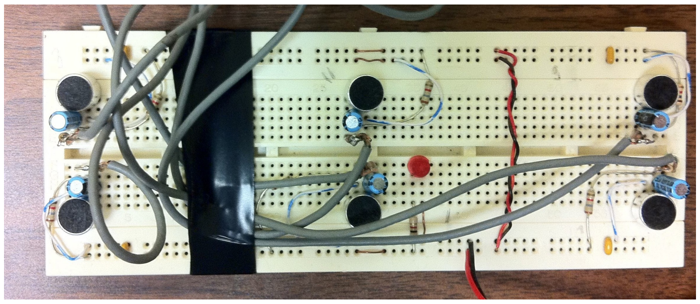


FIG. 3. A 2D array of 6 microphones

(PHAT).

FINDING THE COORDINATES OF THE SOURCE

The location problem in two dimensions, implemented with only two microphones can be represented in the schematic form as shown in Figure 5.

Here, d represents the delay in signals. Starting with two microphones and assuming their coordinates to be (0,0) and (0,s) where s is the distance between them, the time delay d can be represented in the form of the following equation

$$\sqrt{x^2 + (y - s)^2} - \sqrt{x^2 + y^2} = d/v,$$

where v is the velocity of sound in air (approximately 34cm/millisecond). For simplicity, we can normalize to units where $v = 1$ and just drop it from the equation.

Simplifying the equation we get the following equation of a hyperbola,

$$\frac{(y - s/2)^2}{d^2} - \frac{x^2}{s^2 - d^2} = \frac{1}{4}.$$

Each delay value, when simplified and plotted, corresponds to a single hyperbola. As a result, in two dimensions we can find the coordinates of the sound source using three microphones say at (0,-s), (0,0) and (0,s) and then finding the intersection of the corresponding two hyperbolas. As an example, the plot of the two hyperbolae using the values $s = 2$, $d_1 = -1.0804$ and $d_2 = 1.5883$ is shown in Figure 6. There are two intersection points; in this model, we have to use some additional knowledge from the experimental setup to select which of the two identifies the locus of the initial sound source.

A larger number of microphones would result in larger number of delay values and give better accuracy. However, the algebraic method to solve the intersection of more than two hyperbolas becomes complex. The method is simplified if we assume the distance of the sound source from the microphones to be large as compared to the distance between microphones. The hyperbola can then be approximated using the respective asymptote. The equation of the two left and right asymptotes of the hyperbola formed by microphones at $(0, 0)$ and $(0, s)$ are given as follows:

$$y = \frac{s}{2} \pm \frac{d}{\sqrt{d^2 - s^2}} x.$$

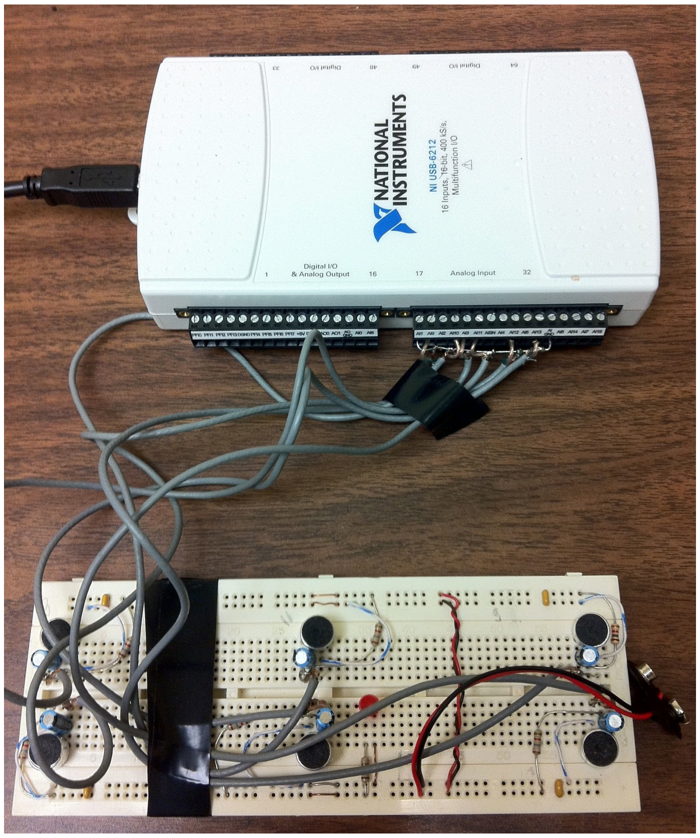


FIG. 4. A 2D array with DSP unit

Using the approximation the method to find the point of intersection becomes a simple linear solution. With several microphones, the system is over-determined so a best estimate is found using the solution to the normal equations. Figure 7 shows the two linear asymptotes for the two hyperbolas arising from two pairs of microphones.

Unfortunately, it increases the error. The same values of s, d_1, d_2 and approximating resulted in an error of 0.0697. This error depends on the values of s, d_1, d_2.

Figure 8 shows the asymptotic lines for several hyperbolas that arise from an array of five microphones. The nearly parallel lines shows that the locus of the sound source is well constrained in one direction, perpendicular to the asymptotes, and not so well constrained in the direction parallel to the asymptotes.

GAUSS-NEWTON, LEAST SQUARES

The error can be reduced using several microphones, combined with the Gauss-Newton method for minimization (see Nocedal and Wright (1999)). The method is used to minimize the nonlinear least squares problem. The residual function, $r(x, y)$ here to be minimized is the sum of squared differences between the delays as predicted at the approximate

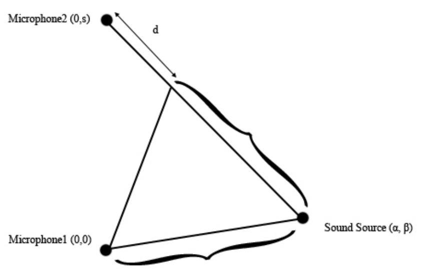


FIG. 5. Time delay in the microphone

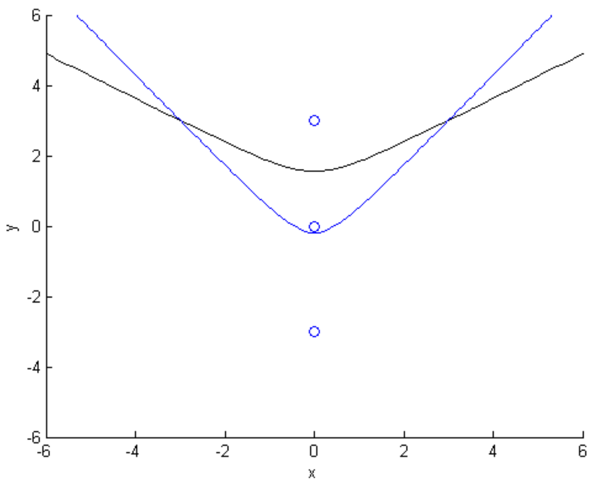


FIG. 6. Two hyperbolae whose intersection locates the event

point (x, y) and the actual recorded delay, $d(i, j)$ between pairs of microphones:

$$r(x,y) = \sum_{i,j} \left(\sqrt{(x-x_i)^2 + (y-y_i)^2} - \sqrt{(x-x_j)^2 + (y-y_j)^2} - d(i,j) \right)^2$$

where indices i, j indicate the enumeration of the microphones, and point (x_i, y_i) is the location of the i-th microphone.

Note it is not necessary to sum over all possible pairs of microphones; a subset may suffice.

This Gauss-Newton minimization method reduces the error to less than 10^{-6} in three iterations for the source coordinates (50,60) with the number of microphones set to 5 and distance between them, s, as 5 units.

EXTENSION TO THREE DIMENSIONS, AND INHOMOGENEOUS MEDIA

Setting the microphones into a linear array helps us solve the problem only in two dimensions. Using one or more microphones on top or bottom of the array all the three

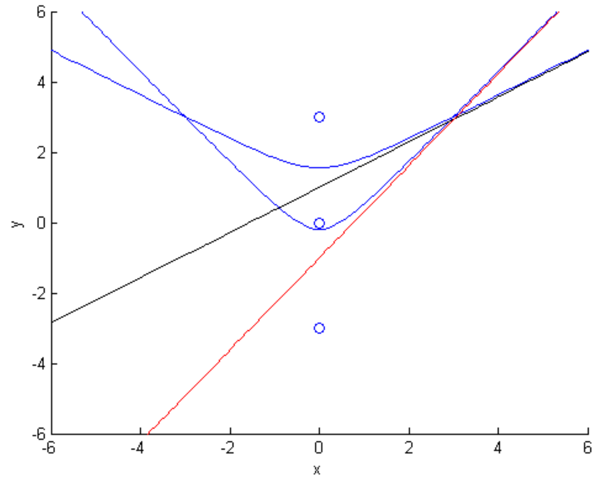


FIG. 7. Approximating the hyperbolae with linear asymptotes

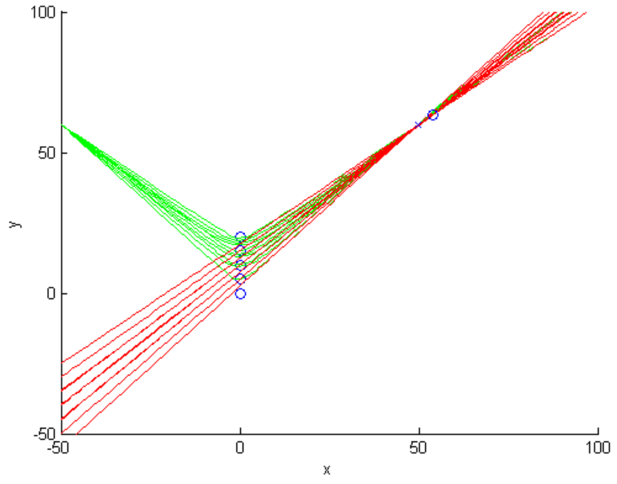


FIG. 8. Several asymptotes, from several microphone pairs

coordinates of the source can be determined.

Our first attempt to solve in three dimensions simply used microphone in a vertical line to solve for two real dimension, then use the microphone in a horizontal line to resolve an additional dimension.

A better approach is to use the least squares approach, minimizing the residual function

$$r(x,y,z) =$$

$$\sum_{i,j}\left(\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}-\sqrt{(x-x_j)^2+(y-y_j)^2+(z-z_j)^2}-d(i,j)\right)^2,$$

where indices i, j indicate the enumeration of the microphones, $d(i, j)$ is the measure travel time delay (converted to a distance), the point (x_i, y_i, z_i) is the location of the i-th microphone in three dimensions, (x, y, z) is the estimated point of the source event. Gauss-Newton is used to find the minimum.

More generally, in the case of heterogenous media where the velocity field is not constant, a ray tracer is used to estimate the time of travel between the source event at position (x, y, z) and the microphone at (x_i, y_i, z_i) to give a time value $T[(x, y, z), (x_i, y_i, z_i)]$. With the measured time delay between microphones as $TD(i, j)$, the residual function to minimize is

$$r(x, y, z) =$$
$$\sum_{i,j} \{T[(x, y, z), (x_i, y_i, z_i)] - T[(x, y, z), (x_j, y_j, z_j)] - TD(i, j)\}^2 .$$

In our physical simulation, the sounds were traveling through air, so there was no need to implement a numerical ray tracer.

FINDING THE DELAY IN SIGNALS USING CROSS CORRELATION

The time delay between received signals can be computed using several methods. One of the most common methods uses the cross correlation of the two received signals and seeks the optimal delay difference to find the peak correlation.

$$C(d) = \sum_t s_1(t) s_2(t - d)$$

$$\text{delay } d = \arg\max[C(d)]$$

where $s_1(t)$ and $s_2(t)$ are the received signals.

We also applied a simple filter to the received signals to reduce the noise and improve the quality of the recorded event.

FINDING THE DELAY USING PHASE TRANSFORM

A review of the literature suggest a phase transform (PHAT) method for computing signal delay should perform better in cases of low SNR in the recorded signal. The PHAT method introduces a weighing function to sharpen the peak of the cross correlation. The weighing function divides the cross spectrum with its own magnitude preserving only the phase information.

$$\begin{aligned} G_{s_1,s_2} &= FFT(crosscorrelation) \\ X &= \frac{G_{s_1,s_2}}{|G_{s_1,s_2}|} \\ C(d) &= FFT^{-1}(X) \\ \text{delay } d &= \arg\max[C(d)] \end{aligned}$$

Our initial experiments indicate the PHAT methods performs significantly better than cross correlation in noisy environments. While this is consistent with what we read in the literature (see Zhang and Abdulla (2005)), it seems somewhat counter-intuitive as one might expect the spectral division to enhance the noise. This is an area for further experimentation.

THE DSP SETUP

We used a National Instruments multichannel analog to digital converter to record signals from the microphones, and processed the signal stream in the LabView software environment. LabView allows the user to create a virtual instrument with custom made algorithms to process the digital signals to whatever specification.

Figure 9 shows a sample of the iconic programming in LabView.

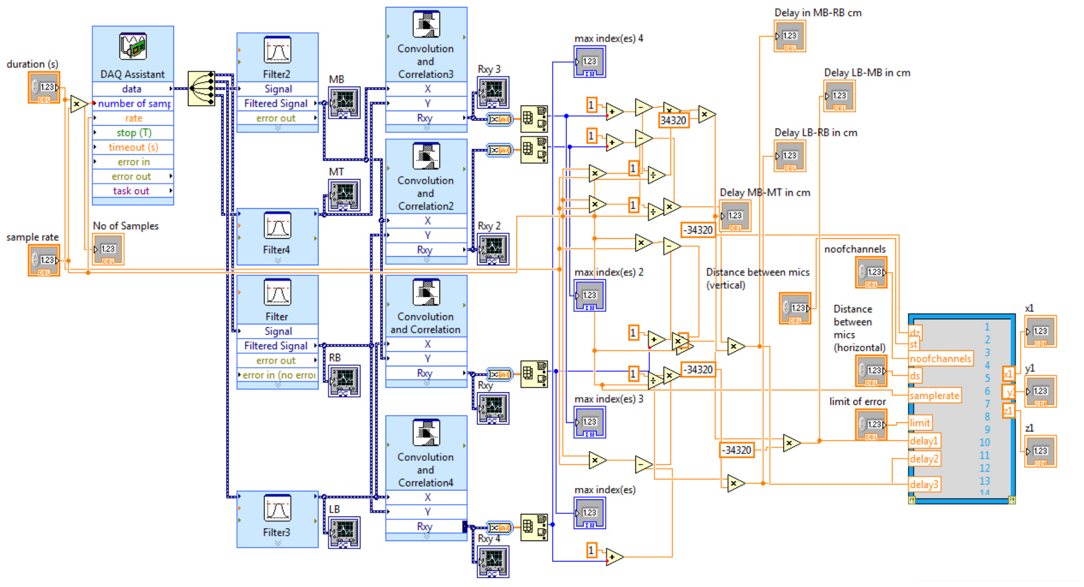


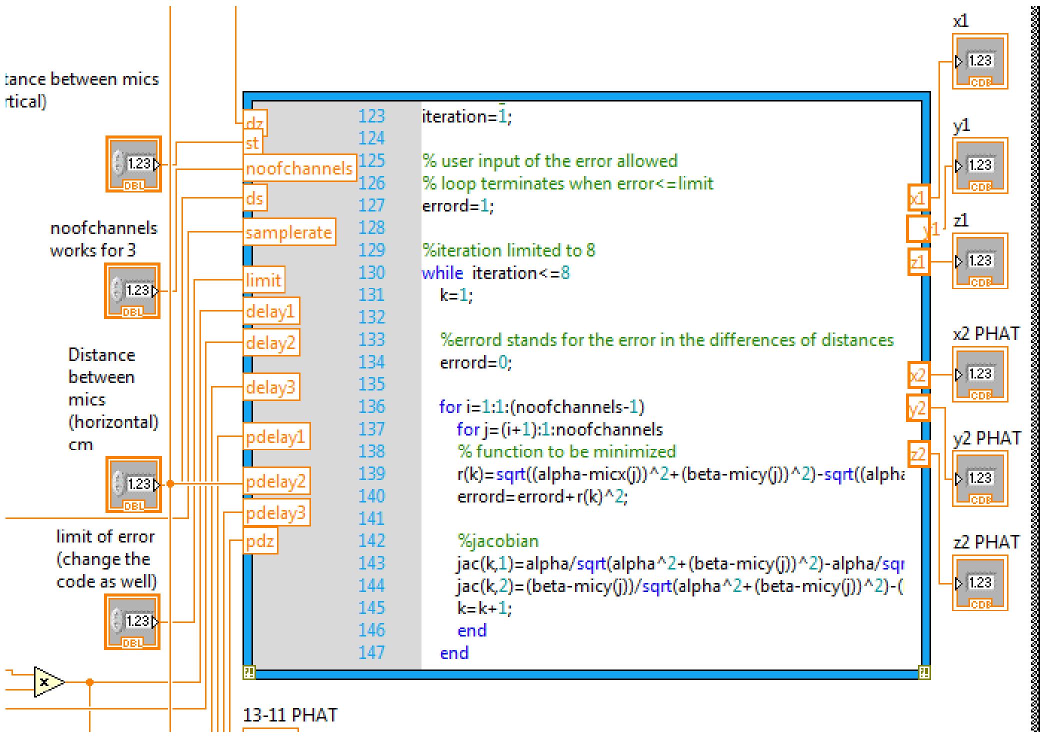
FIG. 9. Iconic programming in LabView

Figure 10 shows some key steps in the event localization algorithm. The signals from the microphone are filtered, then cross-correlated, and the resulting maximum in the cross-correlation is used to compute the relative delay in signal arrival time between two microphones. The relative delays are input into a C-code routine that computes the best approximation to the event location.

EVALUATING THE SYSTEM

Given the limited time of the project (12 weeks in the summer), we were hard pressed to get the system up and running, so only a few tests of accuracy in the locating of events was possible. Some summary points are noted here.

With microphone spacing on the order of 15 centimetres, and events approximately 60 centimetres away, in early tests we were able to reliably obtain accuracies to within a few



```
iteration=1;

% user input of the error allowed
% loop terminates when error<=limit
errord=1;

%iteration limited to 8
while  iteration<=8
   k=1;

   %errord stands for the error in the differences of distances
   errord=0;

   for i=1:1:(noofchannels-1)
      for j=(i+1):1:noofchannels
      % function to be minimized
      r(k)=sqrt((alpha-micx(j))^2+(beta-micy(j))^2)-sqrt((alpha
      errord=errord+r(k)^2;

      %jacobian
      jac(k,1)=alpha/sqrt(alpha^2+(beta-micy(j))^2)-alpha/sqr
      jac(k,2)=(beta-micy(j))/sqrt(alpha^2+(beta-micy(j))^2)-(
      k=k+1;
      end
   end
```

FIG. 10. Sample code in the LabView syste,

centimetres. The method was rather sensitive to echoes, so it was necessary to cover hard surfaces near the microphone array with sound-absorbent materials. It was also sensitive to the type of sound source – finger snaps led to relatively high errors, while a tight snap of a pen cap or metal clip gave better resolution in space. The PHAT method appeared the give better results than cross-correlation in estimating relative time delay. Some experimentation with the DSP processing was necessary, to find appropriate filters and correlation methods that would accurate resolve the sound event and give reliable time delays. With the DSP interfaced to a desktop computer, we were able to obtain essentially real time tracking of events. It would have been nice to get a graphical tracking as well, but there was insufficient time to implement that work.

FUTURE WORK

This simple set up was only sufficient to show that spatial localization of events is possible using a 2D array of microphones. The assumption that events are isolated in time is too restrictive – a more realistic model will have many events occurring at many locations, in close time proximity. Much more sophisticated models will be needed to numerically isolated and locate this events.

The inverse problem method will be implemented, using a 4D model that places multiple events in both space time. Least squares fitting, back projection and filtering will be tested as possible methods to numerically locate the events. By testing in the open laboratory environment, we will be able to evaluate a variety of geometries for microphone layout test for numerical accuracy, and perhaps experiment with inhomogeneous media by inserting baffles and obstacles in the air path. These would lead to more realistic models for the seismic modelling.

CONCLUSIONS

A simple physical model of a microseimic monitoring system was constructed and tested. With single, isolated sound events such as a metal click or snap, we were able to numerically locate the position of the sound event using cross correlation and relative time delays as inputs to a numerical algorithm, Gauss-Newton method was an effective method for finding the most likely location given the recorded data. Further testing is necessary just to verify the numerical accuracy of the method. More complex models in 4D (three space, plus time) and inverse theory methods will be needed to get more realistic methods that work with multiple, clustered sound events that more accurately represent real geophysical situations.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the continuing support of the sponsors of the CREWES and POTSI research consortia, the Mitacs Globalink program, and our funding partners PIMS, NSERC, and Mprime.

REFERENCES

Nocedal, J., and Wright, S., 1999, Numerical Optimization: Springer.

Zhang, Y., and Abdulla, W. H., 2005, A comparative study of time-delay estimation techniques using microphone arrays, School of Engineering Report 619, Dept Electrical and Computer Engineering, University of Auckland.

APPENDIX – CODE

Indicated below is the MATLAB code used to compute the location of acoustic events from the recorded data, as coded by student R. Shukla.

```
for i=1:1:noofchannels
   micx(i)=0;
   micy(i)=(i-1)*ds;
   micz(i)=0;
end
%initializing the arrays used
a=zeros(1,2);
b=zeros(1,2);
c=zeros(1,2);
delta=zeros(1,2);
d=zeros(1,noofchannels);
r=zeros(1,noofchannels);
s=zeros(1,noofchannels);
%getting the delay values
d(1)=delay1;
d(2)=delay2;
d(3)=delay3;
k=1;
for i=1:1:(noofchannels-1)
   for j=(i+1):1:noofchannels
      s(k)=micy(j)-micy(i);
      a(i)=2*d(k);
      b(i)=2*sqrt((s(k))^2-(d(k))^2);
      c(i)=sqrt((s(k))^2-(d(k))^2)*(micy(i)+micy(j));
      k=k+1;
   end
end
%Initial Guess
alpha=(dot(a,c)*dot(b,b)-dot(a,b)*dot(c,b))/(dot(a,a)*dot(b,b)-
dot(a,b)*dot(a,b));
beta=(dot(a,a)*dot(b,c)-dot(a,b)*dot(a,c))/(dot(a,a)*dot(b,b)-
dot(a,b)*dot(a,b));
%% Gauss Newton Method
%counting the iterations
iteration=1;
% user input of the error allowed
% loop terminates when error<=limit
mics=noofchannels;
errord=1;
while iteration<=8
   k=1;
   %errord stands for the error in the differences of distances
   errord=0;
   for i=1:1:(noofchannels-1)
      for j=(i+1):1:noofchannels
      % function to be minimized
      r(k)=sqrt((alpha-micx(j))^2+(beta-micy(j))^2)-sqrt((alpha-micx(i))^2+
      (beta-micy(i))^2)-d(k);
      errord=errord+r(k)^2;
      %jacobian
      jac(k,1)=alpha/sqrt(alpha^2+(beta-micy(j))^2)-alpha/sqrt(alpha^2+(betamicy(i))^2);
      jac(k,2)=(beta-micy(j))/sqrt(alpha^2+(beta-micy(j))^2)-(betamicy(i))/sqrt(alpha^2+(beta-micy(i
      k=k+1;
      end
   end
   %finding the delta
   MATA=jac'*jac;
   MATC=-jac'*r';
   delta=MATA\MATC;
   %adding the delta
   alpha=alpha+delta(1);
   beta=beta+delta(2);
   iteration=iteration+1;
end
%% Real Coordinates
radius=sqrt(alpha^2+((beta-ds)^2));
z1=(radius^2+st^2-((dz-radius)^2))/(2*st);
x1=sqrt(alpha^2-(z1^2));
y1=beta;
%%PHAT METHOD
%initializing the arrays used
a=zeros(1,2);
b=zeros(1,2);
c=zeros(1,2);
delta=zeros(1,2);
d=zeros(1,noofchannels);
r=zeros(1,noofchannels);
s=zeros(1,noofchannels);
%getting the delay values
d(1)=pdelay1;
d(2)=pdelay2;
d(3)=pdelay3;
k=1;
for i=1:1:(noofchannels-1)
   for j=(i+1):1:noofchannels
      s(k)=micy(j)-micy(i);
      a(i)=2*d(k);
      b(i)=2*sqrt((s(k))^2-(d(k))^2);
      c(i)=sqrt((s(k))^2-(d(k))^2)*(micy(i)+micy(j));
      k=k+1;
   end
end
%Initial Guess
alpha=(dot(a,c)*dot(b,b)-dot(a,b)*dot(c,b))/(dot(a,a)*dot(b,b)-
dot(a,b)*dot(a,b));
beta=(dot(a,a)*dot(b,c)-dot(a,b)*dot(a,c))/(dot(a,a)*dot(b,b)-
dot(a,b)*dot(a,b));
%% Gauss Newton Method
%counting the iterations
iteration=1;
% user input of the error allowed
% loop terminates when error<=limit
errord=1;
%iteration limited to 8
while iteration<=8
k=1;
%errord stands for the error in the differences of distances
errord=0;
for i=1:1:(noofchannels-1)
for j=(i+1):1:noofchannels
% function to be minimized
r(k)=sqrt((alpha-micx(j))^2+(beta-micy(j))^2)-sqrt((alpha-micx(i))^2+
(beta-micy(i))^2)-d(k);
errord=errord+r(k)^2;
%jacobian
jac(k,1)=alpha/sqrt(alpha^2+(beta-micy(j))^2)-alpha/sqrt(alpha^2+(betamicy(
i))^2);
jac(k,2)=(beta-micy(j))/sqrt(alpha^2+(beta-micy(j))^2)-(betamicy(
i))/sqrt(alpha^2+(beta-micy(i))^2);
k=k+1;
end
end
%finding the delta
MATA=jac'*jac;
MATC=-jac'*r';
delta=MATA\MATC;
%adding the delta
alpha=alpha+delta(1);
beta=beta+delta(2);
iteration=iteration+1;
end
%% Real Coordinates
radius=sqrt(alpha^2+((beta-ds)^2));
z2=(radius^2+st^2-((pdz-radius)^2))/(2*st);
x2=sqrt(alpha^2-(z2^2));
y2=beta;
hold off;
subplot(2,1,1);
axis([-30 45 -80 10]);
plot(micy,micx,'r:+');
hold on;
plot(y1,-x1,'x');
plot(y2,-x2,'o');
title('Top View');
hold off;
subplot(2,1,2);
axis([-30 45 -30 45]);
plot(micy,micz,'r:+');
hold on;
plot(y1,z1,'x');
plot(y2,z2,'o');
title('Front View');
hold off;
```