A TECHNIQUE FOR ASSESSING SHORT BASELINE ARRAY TILT ERRORS

Robert R. Read

May 1991

Approved for public release; distribution is unlimited.
Prepared for:
Naval Postgraduate School,
Monterey, CA 93955

91-02409
This report was funded by the Naval Postgraduate School Research Program.

This report was prepared by:

Robert R. Read
Professor of Operations Research

Reviewed by:

Released by:

PETER PURDUE
Professor and Chairman
Department of Operations Research

PAUL J. MARTO
Dean of Research
**Title**: A Technique for Assessing Short Baseline Array Tilt Errors

The coherence of three dimensional position locations determined from a splicing together of positions generated by contiguous short baseline tracking arrays is quite sensitive to the orientations of the individual arrays. These orientations can be checked with the use of long baseline methods applied to the signal arrival times at a triplet of sensors, each from a different array, but representing the same sound source. The report presents a sharp algorithm for accomplishing this. It operates in three dimensions and tests out well in the cases attempted.
A TECHNIQUE FOR ASSESSING SHORT BASELINE ARRAY TILT ERRORS

R. R. Read

Abstract

The coherence of three dimensional position locations determined from a splicing together of positions generated by contiguous short baseline tracking arrays is quite sensitive to the orientations of the individual arrays. These orientations can be checked with the use of long baseline methods applied to the signal arrival times at a triplet of sensors, each from a different array, but representing the same sound source. The report presents a sharp algorithm for accomplishing this. It operates in three dimensions and tests out well in the cases attempted.
1. INTRODUCTION

The experimental setting and technical background are contained in references [1, 2]. It is presumed that the reader is familiar with the background. In short, consider three short baseline arrays located, approximately, at the vertices of an equilateral triangle. Each array has four hydrophone receivers, called the x, y, z, and c phones, placed at four of the corners of a cube and forming a (local) Cartesian coordinate system. The x, y, and z array arms (each having length D) are viewed as coordinate axes stemming from an origin at the c phone. The long baseline technique uses one phone from each array. For convenience of presentation we will use only the c phones, but conceptually, it matters not which phone is selected from a particular array.

A single sound source (ping) is received by each of the three c phones and, because of the synchronous nature of the data collection system, we have three sound ray transit times; call them \( t_1, t_2, t_3 \). Also the three position locations (poslocs) are available from the short baseline processing; call them \( x_i, y_i, z_i \) for \( i=1,2,3 \), respectively. These are used to initialize the algorithm. All three of these locations are in the range coordinate system. i.e. the two horizontal components are measured from a common origin and the vertical (z-value) is measured from the water surface, positive downward.

Similarly we have the three entrance angles, \( \theta_1, \theta_2, \) and \( \theta_3 \), each representing the estimated elevation angle of the sound ray at the c phones and measured from the horizontal. The overall goal is to modify these entrance angles in a way that will produce a common vector for the poslocs of the three arrays. (For the present we are ignoring all other sources of error and focusing on the question of array tilt.) The difference between the final and initial entrance angle at a given c phone provides an array tilt correction for the azimuthal direction of the sound source. Two such corrections for two different azimuths can be used to correct the array's two orientation angles, \( X_{TILT} \) and \( Y_{TILT} \). In addition, the azimuth error can be used to adjust \( Z_{ROT} \).
The entrance angle and the transit time serve as the customary inputs to ray tracing methods that return the horizontal distance and the vertical displacement of the sound source measured from the receiver location. (Ray tracing software of both the isospeed and isogradient type are contained in reference [21].) These methods can also be used in inverse fashion. E.g. reference [2] contains a program RAYFIT which takes the positions of the source and receiver as input and returns the transit time and the entrance angle as output. In the present task we use transit time, receiver location, and source depth as input and compute the entrance angle and horizontal range of the source as output.

The locations of the three c phones are known in range coordinates and from these values we can compute \( h_1, h_2, \) and \( h_3 \), the horizontal distances of each phone from the position of the sound source as determined originally by the short baseline methodology. This step is necessary because ray tracing algorithms are two dimensional (range and depth).

This is the setting for the description of the algorithm which is presented in the next section. The results of testing the method are presented in section 3. Section 4 contains some discussion. Fortran source code is contained in an Appendix.

2. METHOD

The algorithm utilizes three dimensional ray tracing in an iterative way until the three poslocs (position locations) agree. The three transit times are held fixed throughout. The first step is to adjust the entrance angles so that the three vertical values are the same. This done we adjust the horizontal ranges \( h_1, h_2, \) and \( h_3 \). These latter values are the radii of the three circles representing the locus of the source if the depth were correct. See Figure 1. In the second step we vary the common depth iteratively until the three circles have a common intersection point. (If there is no common intersection then some other source of error is dominating.)
More specifically our technique for step one is to select one of the three values: \( z_1, z_2, \) and \( z_3; \) determine which array is associated with that value and change the entrance angles at the other two c phones until all three z's are equal to this selected value. Some care must be applied in doing this. It normally follows that the vertical component of a posloc can be reduced and the horizontal shortened by increasing the entrance angle at the phone; decreasing that angle will normally increase the depth and the horizontal. But local aberrations in the sound speed depth velocity profile could cause some mischief and in a given instance there may appear departures from the monotone pattern.

Having and maintaining a common vertical for the three poslocs allows us to proceed as follows. Above we have two versions of the three circles whose radii are the horizontal ranges. The first set represents the case in which the currently determined poslocs are too shallow. The lowering of the common level (i.e. increase the z value) will increase the radii and this can be continued until the circles intersect at a common point. The second set illustrates the case in which the common vertical level is too deep. In this case the radii must be decreased (decrease the z value) until the circles converge. In the former case the entrance angles must be lowered and in the latter they must be raised. An algorithmic description of the process follows.
Algorithm

Setting. A depth velocity sound profile is available for all sound ray tracing type operations. The locations of the three receivers (in range coordinates) are

$$A_i = \begin{cases} a_1(i) \\ a_2(i) \\ a_3(i) \end{cases} \text{ for } i = 1,2,3$$

and the three components are nominally east, north and vertical. The three initial poslocs for the sound source are

$$(x_i, y_i, z_i) \text{ for } i = 1,2,3$$

and the associated transit times and entrance angles are

$$\begin{align*}
   t_i &= \text{transit time} \\
   \theta_i &= \text{entrance angle}
\end{align*} \text{ for } i = 1,2,3.$$  

Step 1. Seek a common vertical; adjust the horizontal ranges.

i. The original ranges are

$$h_i = \sqrt{(x_i - a_1(i))^2 + (y_i - a_2(i))^2} \text{ for } i = 1,2,3.$$  

ii. Compute the horizontal distances $D_{12}, D_{13}$ of the first array from the second and third poslocs. If either of these values are greater than $h_1$, then let $z_0 = \max(z_1, z_2, z_3)$. I.e. this choice is tantamount to a presumption of case 1; all depths need to be lowered. Otherwise let $z_0 = \min(z_1, z_2, z_3)$.

iii. Find the two arrays for which $z_i \neq z_0$ and execute the function NEWLOC for each. This program will return adjusted values for the entrance angle and the horizontal range. These values replace their original counterparts and we are positioned with inputs

$$\theta_i(p) \text{ the adjusted entrance angles}$$
hi(p) the adjusted horizontal ranges

and all \( z_i = z_0 \). The poslocs themselves are adjusted to

conform to \( z_0 \) and the appropriate \( h_i(p) \) with the use of their

original azimuths. That is, the values \( x_i \) and \( y_i \) are updated. In

most of what follows the argument \( p \) is dropped from the

notation.

Caveat. At this point it is assumed that either case 1 or case 2 (see figure) is

operative. It is possible that two circles do not intersect and such a

condition will be discovered when an attempt is made to run the

function CIRCSOLV. If that program returns the "discriminant

negative" message then one must back up and vary the depth

level until the condition is corrected before one can proceed. If

correction is not possible then the input structure must be

declared infeasible and the computations abandoned.

Step 2. Set the convergence tolerance \( \varepsilon > 0 \), say \( \varepsilon = .01 \). Choose two of the

circles. Although any two will serve, the algorithm selects the two

circles whose (current) adjusted poslocs are the closest. The

algorithm then views these arrays, temporarily, as arrays 1 and 2.

(The affixed \( h' \) and \( a'(3) \) refer to the horizontal range and vertical

component of location of the third array in this temporary index

system.) Solve for the two points that they have in common. The

function CIRCSOLV will do this, returning \((u, v)\) for \( j = 1, 2 \). The

intersection that we want is the one that is closest to the

horizontal location of the third array. Thus let

\[
D_1 = \sqrt{(u_1 - a'_1(3))^2 + (v_1 - a'_2(3))^2}
\]

\[
D_2 = \sqrt{(u_2 - a'_1(3))^2 + (v_2 - a'_2(3))^2}
\]

\[
D_3 = \min(D_1, D_2).
\]

Next determine whether the three circles have a common

intersection, or whether case 1 or case 2 is operative.

Let
SIGN = 1  if $D_3 \geq h'_3 + \epsilon$

= -1  if $D_3 \leq h'_3 - \epsilon$

= 0  if $|D_3 - h'_3| < \epsilon$

Notice that we have case 1 if SIGN = 1, case 2 if SIGN = -1, and the three poslocs agree if SIGN = 0. If SIGN = 0 we exit the program. In so doing a new set of azimuths will be determined from the common intersection point. If SIGN \neq 0, set $z_{00} \leftarrow z_0$ and update $z_0$ with the equation

$$z_0 \leftarrow z_{00} + \text{SIGN} \cdot \left| 1 - h'_3/D_3 \right| (a'_3(3) - z_{00}) \cdot \text{GAM}$$

and GAM is a tuning constant to speed convergence. (For our applications GAM = 5 appears to work well.)

Having a new value for $z_0$ we must now execute NEWLOC for all three arrays. Thus use the adjusted horizontal ranges and the original azimuths to compute the adjusted poslocs; return to the beginning of Step 2.

Supporting functions

NEWLOC($\theta$, $t$, $h$, $z$, $z_0$, $\theta(p)$, $h(p)$)

Inputs

$\theta$ = elevation angle
$t$ = transit time
$h$ = horizontal range
$z$ = depth of posloc
$z_0$ = depth goal

Outputs

$\theta(p)$ = adjusted elevation angle
$h(p)$ = adjusted horizontal range

Algorithm

Make the trial adjustment

$$\theta(p) \leftarrow \theta + (z_0 - z)/h.$$
Call a ray tracing program that converts the trial entrance angle \( \theta(p) \) and the fixed transit time \( t \) into new values for \( h \) and \( z \).
Return to * if \( |z - z_0| \geq \epsilon \).
Otherwise set \( h(p) = h \) and exit because of convergence.

QUAD(\( A, B, C, X_1, X_2, ER \)) a quadratic equation solver.

**Inputs**
\( A, B, C \): the coefficients of the quadratic equation.

**Outputs**
\( X_1, X_2 \): the real solutions,
\( D \leftarrow B^2 - 4AC \)
Flag ER if \( D < 0 \)
\( X_1 = \left(-B + \sqrt{D}\right)/2A \)
\( X_2 = \left(-B - \sqrt{D}\right)/2A \)

CIRCSOLV(\( A_1, A_2, h_1, h_2, u_1, u_2, v_1, v_2 \))

**Inputs**
\( A_1 = \begin{bmatrix} a_1(1) \\ a_1(2) \end{bmatrix} \) east coordinates of the two circles
\( A_2 = \begin{bmatrix} a_2(1) \\ a_2(2) \end{bmatrix} \) north coordinates of the two circles
\( h_1, h_2 \) circle radii

**Outputs**
\((u_1, v_1)\) first intersection point
\((u_2, v_2)\) second intersection point.

**Code**
\( B_0 \leftarrow h_1^2 - h_2^2 + a_1^2(2) - a_1^2(1) + a_2^2(2) - a_2^2(1) \)
\( B_1 \leftarrow 2(a_1(2) - a_1(1)) \)
\( B_2 \leftarrow 2(a_2(2) - a_2(1)) \)
\( B_1 \leftarrow -B_1/B_2 \quad ; \quad B_0 \leftarrow B_0/B_2 \)
Let

\[ P_1 = 1 + B_1^2 \]
\[ P_2 = 2B_1(B_0 - a_2(1)) - 2a_1(1) \]
\[ P_3 = a_1^2(1) + (B_0 - a_2(1))^2 - h_1^2 \]

Call QUAD \((P_1, P_2, P_3, u_1, u_2, ERR)\)

Then set

\[ u_2 \leftarrow B_0 + B_1u_1 \]
\[ u_2 \leftarrow B_0 + B_1u_1 \]

end.

**Algebraic description of the solution for the intersection of two circles.**

Given the two circles

\[ (x-a_1(1))^2 + (y-a_2(1))^2 = h_1^2 \]
\[ (x-a_1(2))^2 + (y-a_2(2))^2 = h_2^2 \]

We begin by expanding the left hand sides and then subtract the second from the first. This produces

\[ 2x[a_1(2) - a_1(1)] + 2y[a_2(2) - a_2(1)] = h_1^2 - h_2^2 + [a_1^2(2) - a_1^2(1)] + [a_2^2(2) - a_2^2(1)] \]

or

\[ B_1x + B_2y = B_0 \]

and \( B_0, B_1, \) and \( B_2 \) are defined by identification above. If we redefine \( B_1 \leftarrow -B_1/B_2 \) and \( B_0 \leftarrow B_0/B_2 \) then the linear relationship immediately above may be re-expressed as

\[ y = B_0 + B_1x \]

and this may be substituted for \( y \) in the original first circle. Thus
\[(x-a_1(1))^2 + [B_0 + B_1 x - a_2(1)]^2 = h_1^2\]

which is a second order expression in \(x\). In standard form it is

\[P_1 x^2 + P_2 x + P_3\]

where

\[P_1 = 1 + B_1^2\]

\[P_2 = 2B_1[B_0-a_2(1)]-2a_1(1)\]

\[P_3 = a_1^2(1)+[B_0-a_2(1)]^2-h_1^2\]

If the solution of this quadratic is real, then the circles intersect and the two solutions are the \(x\)-components of those two vectors. The \(y\)-components are found from

\[y = B_0 + B_1 x.\]

3. TESTING THE ALGORITHM

This section documents the testing of the algorithm using the six triple crossover sets available. The first five cases are for 10 May 1989 and the sixth for 6 June 1989. (They are identified as cases 1, 3, 4, 5, 6, 7; the original case 2 is defective. The originals can be found in Ref. [1], pp. 14-17.) Graphs of the DV profiles can be found in Ref. [2].

The particulars of testing are described below. A filtered central posloc is developed (data are averaged) for each array in a triple overlap pointset count. The transit time to that position is generated by our rayfitting algorithm. Then long baseline methods are used to produce the (consensus) posloc. We
also produce the azimuth and elevation angles used for all poslocs. From these we get ZROT corrections and tilt correction for the particular azimuth. Azimuth is measured counter clockwise from the east.

**Remark:** This methodology assumes that the array c-phone locations are correct. We used Table B.2 (corrected) of Ref. [2]. Angles are measured in radians unless otherwise specified.

<table>
<thead>
<tr>
<th>Case 1 (5/10/89)</th>
<th>ARRAY 1</th>
<th>ARRAY 2</th>
<th>ARRAY 11</th>
<th>LONGBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>posloc</td>
<td>23195.5</td>
<td>23155.2</td>
<td>23172.2</td>
<td>23180.4</td>
</tr>
<tr>
<td></td>
<td>-1927.8</td>
<td>-1940.2</td>
<td>-1915.4</td>
<td>-1934.5</td>
</tr>
<tr>
<td></td>
<td>-326.4</td>
<td>-382.1</td>
<td>-364.9</td>
<td>-283.4</td>
</tr>
<tr>
<td>azimuth</td>
<td>-0.439507</td>
<td>-2.69574</td>
<td>1.56684</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>-0.442562</td>
<td>-2.694418</td>
<td>1.565204</td>
<td></td>
</tr>
<tr>
<td>horizontal error</td>
<td>16.5</td>
<td>25.8</td>
<td>20.8</td>
<td></td>
</tr>
<tr>
<td>elevation</td>
<td>0.223916</td>
<td>0.209958</td>
<td>0.19348</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>0.234378</td>
<td>0.233427</td>
<td>0.211358</td>
<td></td>
</tr>
<tr>
<td>error (degrees)</td>
<td>0.6°</td>
<td>1.3°</td>
<td>1.0°</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 3 (5/10/89)</th>
<th>ARRAY 1</th>
<th>ARRAY 2</th>
<th>ARRAY 11</th>
<th>LONGBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>posloc</td>
<td>23101.0</td>
<td>23061.2</td>
<td>23076.8</td>
<td>23085.6</td>
</tr>
<tr>
<td></td>
<td>-1938.2</td>
<td>-1949.5</td>
<td>-1928.8</td>
<td>-1944.9</td>
</tr>
<tr>
<td></td>
<td>-325.2</td>
<td>-380.8</td>
<td>-352.8</td>
<td>-283.7</td>
</tr>
<tr>
<td>azimuth</td>
<td>-0.451718</td>
<td>-2.703184</td>
<td>1.587809</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>-0.454869</td>
<td>-2.701741</td>
<td>1.585941</td>
<td></td>
</tr>
<tr>
<td>horizontal error</td>
<td>16.8</td>
<td>24.8</td>
<td>18.3</td>
<td></td>
</tr>
<tr>
<td>elevation</td>
<td>0.228946</td>
<td>0.205557</td>
<td>0.196589</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>0.239260</td>
<td>0.228152</td>
<td>0.211792</td>
<td></td>
</tr>
<tr>
<td>error (degrees)</td>
<td>0.6°</td>
<td>1.3°</td>
<td>0.9°</td>
<td></td>
</tr>
<tr>
<td>Case 4 (5/10/89)</td>
<td>ARRAY 5</td>
<td>ARRAY 6</td>
<td>ARRAY 15</td>
<td>LONGBASE</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>posloc</td>
<td>53014.7</td>
<td>52981.2</td>
<td>53009.1</td>
<td>53011.3</td>
</tr>
<tr>
<td></td>
<td>-2185.7</td>
<td>-2144.5</td>
<td>-2155.8</td>
<td>-2165.7</td>
</tr>
<tr>
<td></td>
<td>-381.2</td>
<td>-361.0</td>
<td>-367.9</td>
<td>-327.6</td>
</tr>
<tr>
<td>azimuth (original)</td>
<td>-.552438</td>
<td>-.655442</td>
<td>1.623004</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>-.548752</td>
<td>-.649183</td>
<td>1.622606</td>
<td></td>
</tr>
<tr>
<td>horizontal error (ft)</td>
<td>20.3</td>
<td>36.8</td>
<td>10.1</td>
<td></td>
</tr>
<tr>
<td>elevation (original)</td>
<td>.212758</td>
<td>.197559</td>
<td>.212733</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>.225691</td>
<td>.204975</td>
<td>.222314</td>
<td></td>
</tr>
<tr>
<td>error (degrees)</td>
<td>0.7°</td>
<td>0.4°</td>
<td>0.5°</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case 5 (5/10/89)</th>
<th>ARRAY 16</th>
<th>ARRAY 17</th>
<th>ARRAY 26</th>
<th>LONGBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>posloc</td>
<td>64676.0</td>
<td>64642.6</td>
<td>64637.4</td>
<td>64653.3</td>
</tr>
<tr>
<td></td>
<td>-8633.0</td>
<td>-8657.7</td>
<td>-8647.4</td>
<td>-8654.7</td>
</tr>
<tr>
<td></td>
<td>-351.0</td>
<td>-359.5</td>
<td>-343.8</td>
<td>-316.8</td>
</tr>
<tr>
<td>azimuth (original)</td>
<td>-.541822</td>
<td>-2.598895</td>
<td>1.529189</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>-.548626</td>
<td>-2.598187</td>
<td>1.525426</td>
<td></td>
</tr>
<tr>
<td>horizontal error (ft)</td>
<td>31.4</td>
<td>29.3</td>
<td>17.5</td>
<td></td>
</tr>
<tr>
<td>elevation (original)</td>
<td>.202977</td>
<td>.216093</td>
<td>.197164</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td>.210674</td>
<td>.226356</td>
<td>.203495</td>
<td></td>
</tr>
<tr>
<td>error (degrees)</td>
<td>0.4°</td>
<td>0.6°</td>
<td>0.4°</td>
<td></td>
</tr>
</tbody>
</table>
### Case 6 (5/10/89)

<table>
<thead>
<tr>
<th></th>
<th>ARRAY 3</th>
<th>ARRAY 4</th>
<th>ARRAY 54</th>
<th>LONGBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>posloc</td>
<td>38826.5</td>
<td>38794.9</td>
<td>38828.6</td>
<td>38811.5</td>
</tr>
<tr>
<td></td>
<td>915.3</td>
<td>939.9</td>
<td>954.0</td>
<td>951.1</td>
</tr>
<tr>
<td></td>
<td>-349.3</td>
<td>-363.4</td>
<td>-320.1</td>
<td>-322.3</td>
</tr>
<tr>
<td>azimuth</td>
<td>.223151</td>
<td>2.842790</td>
<td>-1.393582</td>
<td></td>
</tr>
<tr>
<td>(original)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.231813</td>
<td>2.839161</td>
<td>-1.397459</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>horizontal error (ft)</td>
<td>38.8</td>
<td>20.0</td>
<td>16.8</td>
<td></td>
</tr>
<tr>
<td>elevation</td>
<td>.207200</td>
<td>.269114</td>
<td>.183713</td>
<td></td>
</tr>
<tr>
<td>(original)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.213329</td>
<td>.281344</td>
<td>.183226</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>0.4°</td>
<td>0.7°</td>
<td>-0.03°</td>
<td></td>
</tr>
<tr>
<td>(degrees)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Case 7 (6/6/89)

<table>
<thead>
<tr>
<th></th>
<th>ARRAY 1</th>
<th>ARRAY 2</th>
<th>ARRAY 11</th>
<th>LONGBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>posloc</td>
<td>23307.2</td>
<td>23265.2</td>
<td>23274.3</td>
<td>23288.6</td>
</tr>
<tr>
<td></td>
<td>-2302.1</td>
<td>-2309.4</td>
<td>-2286.9</td>
<td>-2304.1</td>
</tr>
<tr>
<td></td>
<td>-363.5</td>
<td>-392.5</td>
<td>-366.2</td>
<td>-298.7</td>
</tr>
<tr>
<td>azimuth</td>
<td>-.505522</td>
<td>-2.607256</td>
<td>1.542357</td>
<td></td>
</tr>
<tr>
<td>(original)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-.507964</td>
<td>-2.605552</td>
<td>1.538843</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>horizontal error (ft)</td>
<td>18.7</td>
<td>24.0</td>
<td>22.4</td>
<td></td>
</tr>
<tr>
<td>elevation</td>
<td>.201174</td>
<td>.203625</td>
<td>.211814</td>
<td></td>
</tr>
<tr>
<td>(original)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.216092</td>
<td>.225676</td>
<td>.228084</td>
<td></td>
</tr>
<tr>
<td>(longbase)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>0.9°</td>
<td>1.3°</td>
<td>0.9°</td>
<td></td>
</tr>
<tr>
<td>(degrees)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The results are rather dramatic, but cannot be taken too literally. The set of original poslocs were generated by the processing that is currently in use. Reference [2] contains an error study of this processing. It identifies some sources of systematic error including the method of initializing the ray tracing algorithm, the use of 25 ft. water layer increments, and isospeed ray tracing. The presence of such systematic errors can mask the true effect. Recall that the original poslocs in our data are averages of those poslocs in the original triple overlap region; the transit times are computed by rayfitting with five feet water layer increments and isogradient ray tracing.

The effect is illustrated in Figure 2, which treats case 5 but is representative of all the cases. The original transit times for 13 point counts in the triple overlap region of arrays 16, 17, and 26 are plotted vs. point count. The connecting traces serve to show smoothness in the original transit times. The original poslocs are rather variable. They were averaged (as were the point count numbers) to produce stability. The transit times fitted to the three averages are marked with bold gray circles in the figure. They appear

![Figure 2. Transit times to C-Hydrophones](attachment:figure2.png)
over the average point count value of 8109.3. It is clear that they do not fall on the smooth traces. Thus we have some systematic error in our angle and other assessments. The systematic errors identified in Ref. [2] surely contribute to this condition.

**Remarks:** The algorithm seems to work well. One should note that the effects of small errors are quite remarkable. A one-half degree error in one of the horizontal arrays arms represents about 3 inches (since the arm is 30 feet long). If the posloc is in an overlap area (say, horizontal range > 4000 feet) then the corresponding vertical posloc error is about 40 feet.

Three of the six cases involve the array triple (1, 2, 11). These cases are independent, i.e., the point count sets are quite different for cases 1 and 3; and a different date for case 7. It is encouraging to note that the computed tilt error values are consistent. This supports the previously stated concern about the effects of systematic errors.

A second independent estimate of the rotation error obtained from differing (azimuth) directions may or may not be consistent. If not, it may provide information about the array locations. But consistency is necessary to proceed with tilt corrections.

**4. SUMMARY**

The algorithm appears to work quite well. In all cases the convergence was rapid. In no instance was the discriminant negative. Indeed a trap for that caveat has not yet been written into the code.

As noted, the particular results themselves should not be taken too literally. The systematic errors identified for the current processing system are the most likely suspect areas. Uncertainty in the array locations is the next concern. The use of filtered poslocs in the test cases also may be contributing to some deception. These values are the centroids of the local data sets
determined by a biased processing system. Other sources of bias may be operating, e.g., drifts in the depth-sound velocity profile.

The present work is concerned only with the development and testing of the algorithm. The next step calls for some sensitivity testing. It is important to learn how the method is affected by the error sources mentioned. There is also the question of how the tool should be used. E.g., should the poslocs be filtered before or after its application. Once the behavior is well understood, we can turn to the business of designing an experiment for calibrating the array tilts.
APPENDIX. PROGRAM LONGBASE

PROGRAM LONGBASE

12/19/90

LONGBASE.FOR is a long base line POSLOC method for the triple crossover data from the Nanoose range.

A1,A2,A3 are input vectors providing locations in 3-D for the three involved C-Hydrophones. Specifically

A1(I) are the downrange (X) coordinates.
A2(I) are the crossrange (Y) coordinates.
A3(I) are the depth (Z) coordinates. (Depth is positive down orientation).

DIMENSION A1(3),A2(3),A3(3),X(3),Y(3),Z(3),TH0(3),T(3)
DIMENSION H(3),AR1(2),AR2(2),HP(3),THP(3),PH0(3),PH1(3)
DIMENSION XX(3),YY(3),TH00(3),AR(4)
COMMON /SETI/ L(300),VEL(300),V0(300),V1(300),DZ(300),
*  LM(300)
REAL*8 L,VEL,V0,V1,DZ,LM
REAL*8 A1,A2,A3,X,Y,Z,DZ0,AA1,AA2,P1,P2,TH0,T,Z0,Z00,H
REAL*8 D12,D13,D23,AR1,AR2,D1,D2,D3,HP,THP,PH0,PH1,TH00
REAL*8 VA,VB,UA,UB,Z1,Z2,Z3,TH1,TH00,XX,YY,RDH,DS,DM,GAM

INTEGER*4 DL1,DL2,DH,MARR,SIGN,AR,ERR
CHARACTER*15 DVNAME,OUTNAM
CHARACTER*1 RES

5  EPS = 1D-3
ERR = 0

C GAM is a convergence tuning constant. Adjustment may speed convergence.
GAM = 5.0D0

C Load DVT information. Adjust depth to be positive downward.
WRITE(*,'(A)') 'ENTER VELOCITY PROFILE:'
READ(*,230)DVNAME
OPEN(UNIT=2,FILE=DVNAME,STATUS = 'OLD')
I = 1
10 READ(2,220,ERR=20,END=30)L(I),VEL(I)
L(I) = -L(I)
I = I + 1
GOTO 10
20 WRITE(*,*) 'THERE WAS AN ERROR READING THE DVT FILE !'
STOP
30 CLOSE(UNIT=2)
M = I-1

C Prepare for isogradient ray tracing.
C Form the set of layer midpoints
 DO 40 I = 1,M-1
 40 LM(I) = .5*(L(I) + L(I+1))

C Form depth increments, and all sound velocity slopes and intercepts.
 DO 50 I = 1,M-2
 50 DZ(I) = LM(I+1) - LM(I)
    V0(I) = (LM(I+1)*VEL(I) - LM(I)*VEL(I+1))/DZ(I)
    V1(I) = (VEL(I+1) - VEL(I))/DZ(I)
 CONTINUE

LM(M) = LM(M-1) + DZ(M-2)

C Enter the appropriate values for the data runs.
 WRITE(*,'(A)') ' ENTER OUTPUT FILE NAME:'
 READ(*,230)OUTNAM
 OPEN(UNIT=3,FILE=OUTNAM,STATUS='NEW')
 WRITE(*,200)

C Input the C-Hydrophone location values.
 WRITE(*,'(A)') ' ENTER THE NUMBER OF THE 1ST ARRAY:'
 READ(*,*)AR(1)
 CALL ARRAY(AR(1),A1(1),A2(1),A3(1),ERR)
 WRITE(*,*)
 WRITE(*,'(A)') ' ENTER THE NUMBER OF THE 2ND ARRAY:'
 READ(*,*)AR(2)
 CALL ARRAY(AR(2),A1(2),A2(2),A3(2),ERR)
 WRITE(*,*)
 WRITE(*,'(A)') ' ENTER THE NUMBER OF THE 3RD ARRAY:'
 READ(*,*)AR(3)
 CALL ARRAY(AR(3),A1(3),A2(3),A3(3),ERR)
 IF(ERR.EQ.1) THEN
  WRITE(*,'(A)') '! THERE WAS AN ERROR READING THE ARRAYS!'
  WRITE(*,*) '! PLEASE TRY AGAIN!'
  ERR = 0
  GOTO 55
 ENDIF
 ERR = 0
 WRITE(*,*)

C Input initial POSLOC values.
 57 WRITE(*,'(A)') ' WHICH CASE NUMBER DO YOU WANT TO RUN ? (1-7):'
 READ(*,*)AR(4)
 CALL POSLOC(X,Y,Z,AR(4))
 58 WRITE(*,270)X(1),Y(1),Z(1)
 WRITE(*,271)X(2),Y(2),Z(2)
 WRITE(*,272)X(3),Y(3),Z(3)
 WRITE(*,*)
 WRITE(*,'(A)') ' ARE THESE VALUES CORRECT ? (Y/N):'
 READ(*,230)RES
WRITE(*,*),' ' IF((RES.EQ.'N').OR.(RES.EQ.'n')) THEN WRITE(*,'(A)'),' WOULD YOU LIKE TO ENTER THEM ? (Y/N):' READ(*,230)RES WRITE(*,*),' ' IF((RES.EQ.'y').OR.(RES.EQ.'Y')) THEN WRITE(*,'(A)'),' ENTER THE 1ST POSLOC (X,Y,Z):' READ(*,*)X(1),Y(1),Z(1) WRITE(*,'(A)'),' ENTER THE 2ND POSLOC (X,Y,Z):' READ(*,*)X(2),Y(2),Z(2) WRITE(*,'(A)'),' ENTER THE 3RD POSLOC (X,Y,Z):' READ(*,*)X(3),Y(3),Z(3) GOTO 58 ELSE GOTO 57 ENDIF ENDIF

C Run the RAYFIT program to each of the 3 average location C values, and return the transit times T(I), and the entrance angles THO(I).

WRITE(*,*),' ' WRITE(*,*),' ! PROCESSING !' WRITE(*,*),' ' DO 60 I = 1,3

AA1 = 0.0D0 AA2 = A3(I)
P1 = DSQRT((X(I)-A1(I))**2 + (Y(I)-A2(I))**2)
P2 = Z(I)
IEST = 0

CALL RAYFIT1(AA1,AA2,P1,P2,M,VEL,LM,DZ,V0,V1, * T(I),THO(I),TH1,IEST) THO0(I) = THO(I)

60 CONTINUE

C Compute the max dz.
DZ0 = MAX(DABS(Z(I)-Z(2)),DABS(Z(1)-Z(3)),DABS(Z(2)-Z(3)))

C Determine horizontal ranges and azimuths.
DO 70 I = 1,3
H(I) = DSQRT((X(I)-A1(I))**2 + (Y(I)-A2(I))**2)
PHO(I) = DATAN2(Y(I)-A2(I),X(I)-A1(I))
70 CONTINUE

WRITE(3,280)AR(1),AR(2),AR(3),AR(4)
WRITE(3,250)A1(1),A2(1),A3(1)
WRITE(3,251)A1(2),A2(2),A3(2)
WRITE(3,252)A1(3),A2(3),A3(3)
WRITE(3,270)X(1),Y(1),Z(1)
WRITE(3,271)X(2),Y(2),Z(2)
WRITE(3,272)X(3),Y(3),Z(3)
WRITE(3,*)' ' 
WRITE(3,253)H(1),H(2),H(3)

C Begin step 1.
C Choose the initial common depth ZO.
   D12 = DSQRT((X(2)-A1(1))**2 + (Y(2)-A2(1))**2)
   D13 = DSQRT((X(3)-A1(1))**2 + (Y(3)-A2(1))**2)
   IF ((D12.GE.H(1)).OR.(D13.GE.H(1))) THEN
       ZO = MAX(Z(1),Z(2),Z(3))
   ELSE
       ZO = MIN(Z(1),Z(2),Z(3))
   ENDIF
C Determine which array POSLOC has depth ZO.
   DO 75 I = 1,3
      IF(Z(I).EQ.ZO) MARR = I
      WRITE(3,254)DS,ZO
   GOTO 720
   75
   80 D1 = DSQRT((UA-A1(DH))**2 + (VA-A2(DH))**2)
   D2 = DSQRT((UB-A1(DH))**2 + (VB-A2(DH))**2)
   D3 = MIN(D1,D2)
C Test for the confluence of the three circle intersections.
   IF(DABS(H(DH) - D3).LT.EPS) GOTO, 100
C Perform the next iteration.
   Z00 = ZO
   Z0 = Z00 + (A3(DH) - Z00)*(1 - H(DH)/D3)*GAM
   DS = D3 - H(DH)
   WRITE(3,254)DS,Z0
   WRITE(3,253)H(1),H(2),H(3)
   720 CONTINUE
   DO 90 I = 1,3
      IF(I.EQ.MARR) GOTO 90
      CALL NEWLOC(0.0D0,A3(I),M,THO(I),T(I),H(),Z(),ZO)
   90 CONTINUE
C Step 2 control transfer.
   IF(MARR.EQ.4) GOTO 98
C Adjust POSLOCs for new horizontal ranges.
   DO 95 I = 1,3
      IF(I.EQ.MARR) THEN
         XX(I) = X(I)
         YY(I) = Y(I)
      ELSE
      ENDIF
   95 CONTINUE

20
\[
XX(I) = A1(I) + H(I) \cdot DCOS(PH0(I)) \\
YY(I) = A2(I) + H(I) \cdot DSIN(PH0(I))
\]

ENDIF

95 CONTINUE

C Find the two closest POSLOCS.

\[
D12 = DSQRT((XX(1) - XX(2))^2 + (YY(1) - YY(2))^2) \\
D13 = DSQRT((XX(1) - XX(3))^2 + (YY(1) - YY(3))^2) \\
D23 = DSQRT((XX(2) - XX(3))^2 + (YY(2) - YY(3))^2) \\
DM = MIN(D12,D13,D23)
\]

IF(DM.EQ.D23) THEN

DL1 = 2
DL2 = 3
DH = 1

ELSEIF(DM.EQ.D13) THEN

DL1 = 1
DL2 = 3
DH = 2

ELSE

DL1 = 1
DL2 = 2
DH = 3

ENDIF

WRITE(3,255)DH,GAM

C Step 1 completed. Set control for step 2 operations only.

MARR = 4
AR1(1) = A1(DL1)
AR2(1) = A2(DL1)
AR1(2) = A1(DL2)
AR2(2) = A2(DL2)

C Find the next set of circles. Transfer to the test for confluence.

98 CALL CIRCSOLV(AR1,AR2,H(DL1),H(DL2),UA,VA,UB,VB)
GOTO 80

100 CONTINUE

C Finalize after convergence.

IF(D3.EQ.D2) THEN

VA = VB
UA = UB

ENDIF

WRITE(3,260)UA,VA,Z0

C Compute new azimuths.

DO 110 I = 1,3

PH0(I) = DATAN2(Y(I)-A2(I),X(I)-A1(I))

PH1(I) = DATAN2(VA-A2(I),UA-A1(I))

WRITE(3,261)AR(I),PH0(I),PH1(I)

WRITE(3,262)TH00(I),TH0(I)

21
CONTINUE
CLOSE(UNIT=3)
WRITE(*,'(A)')' WOULD YOU LIKE TO RUN ANOTHER CASE ? (Y/N):
READ(*,230)RES
WRITE(*,*)'
IF((RES.EQ.'Y').OR.(RES.EQ.'y')) GOTO 5
STOP

200 FORMAT('DEFINe THE LOCATION OF THE C-HYDROPHONE FOR EACH',
* 'OF THE ARRAYS OF', 'THE TRIPLE OVERLAP. PLEASE',
* 'NOTE THAT WE USE A POSITIVE DOWN', 'ORIENTATION',
* 'FOR DEPTH.')
220 FORMAT(5X,D8.2,5X,D7.2)
230 FORMAT(A)
250 FORMAT(/10X,'A1(1)=',F10.2,' A2(1)=',F10.2,' A3(1)=',F10.2)
251 FORMAT(10X,'A1(2)=',F10.2,' A2(2)=',F10.2,' A3(2)=',F10.2)
252 FORMAT(10X,'A1(3)=',F10.2,' A2(3)=',F10.2,' A3(3)=',F10.2)
253 FORMAT(10X,'H(1)=',F11.4,' H(2)=',F11.4,' H(3)=',F11.4)
254 FORMAT(10X,'DS =',F11.4,' Z0 =',F11.4)
255 FORMAT(/10X,'DH = ',F12,' GAM = ',F4.1)
260 FORMAT(10X,'U =',F10.2,' V =',F10.2,' Z0 =',
* F10.2)
261 FORMAT(10X,'ARR #',F12,' PH0 = ',F10.6,' PH1 = ',F10.6)
262 FORMAT(10X,'TH0 = ',F10.6)
270 FORMAT(10X,'X(1) =',F10.2,' Y(1) =',F10.2,' Z(1) =',F10.2)
271 FORMAT(10X,'X(2) =',F10.2,' Y(2) =',F10.2,' Z(2) =',F10.2)
272 FORMAT(10X,'X(3) =',F10.2,' Y(3) =',F10.2,' Z(3) =',F10.2)
280 FORMAT(10X,'ARRAYS ',F12,' ',F12,' and ',F12,7X,'CASE ',F12)
END

SUBROUTINE NEWLOC(A1,A2,M,TH0,T,H0,Z,Z0)

INPUTS:
A1: HORIZONTAL COORDINATE OF SENSOR
A2: VERTICAL COORDINATE OF SENSOR, POSITIVE DOWN
M: INDEX OF DEEPEST LAYER USED
TH0: ELEVATION ANGLE
T: TRANSIT TIME
H0: HORIZONTAL RANGE
Z: DEPTH OF POSLOC
Z0: DEPTH GOAL

OUTPUTS:
THP: ADJUSTED ELEVATION ANGLE
HP: ADJUSTED HORIZONTAL RANGE
COMMON /SET1/ L(300),VEL(300),V0(300),V1(300),DZ(300),
    LM(300)
REAL*8 L,VEL,V0,V1,DZ,LM
REAL*8 TH0,T,H0,Z,Z0,THP,HP,R,EPS,A1,A2,TH1

C Set the convergence criteria.
EPS = .0010D0
C Compute the slant range
DO 5 J = 2,M
5 CONTINUE
THP = TH0
10 CONTINUE
THP = THP - (Z0-Z)/H0
CALL ISOGRAD1(A1,A2,T,THP,N,LM,VEL,V0,V1,DZ,H0,Z,TH1)
IF (DABS(Z-Z0).GE.EPS) GOTO 10
TH0 = THP
RETURN
END

SUBROUTINE CIRCSOLV(A1,A2,H1,H2,U1,V1,U2,V2)

Determines the two intersection points of two circles.

INPUTS:
   A1: EAST COORDINATE OF THE TWO CIRCLES
   A2: NORTH COORDINATE OF THE TWO CIRCLES
   H1,H2: RADIUS

OUTPUTS:
   U1,V1: 1ST INTERSECTION POINT
   U2,V2: 2ND INTERSECTION POINT

DIMENSION A1(2),A2(2)
REAL*8 A1,A2,B0,B1,H1,H2,U1,U2,V1,V2,P1,P2,P3,B,B2
INTEGER*4 ERR

ERR = 0
B0 = H1**2 - H2**2 + A1(2)**2 - A1(1)**2 + A2(2)**2 - A2(1)**2
B1 = 2*(A1(2) - A1(1))
B2 = 2*(A2(2) - A2(1))
B1 = -B1/B2
B0 = B0/B2
P1 = 1+B1**2
P2 = 2*B1*(B0-A2(1))-2*A1(1)
P3 = A1(1)**2 + (B0-A2(1))**2 - H1**2
CALL QUAD(P1,P2,P3,U1,U2,ERR)
IF (ERR.EQ.1) THEN
WRITE(*,*) 'THERE WAS AN ERROR IN QUAD SOLUTION'
STOP
ENDIF
V1 = B0 + B1*U1
V2 = B0 + B1*U2
RETURN
END

SUBROUTINE QUAD(A,B,C,X1,X2,ERR)

C Solves for the real roots of the quadratic equation.

REAL*8 A,B,C,D,X1,X2
INTEGER*4 ERR

D = B**2 - 4*A*C
IF (D.LT.0.0D0) THEN
ERR = 1
GOTO 10
ENDIF
X1 = (-B + DSQRT(D))/(2*A)
X2 = (-B - DSQRT(D))/(2*A)
10 RETURN
END
REFERENCES


<table>
<thead>
<tr>
<th>Distribution List</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library (Code 0142) Naval Postgraduate School Monterey, CA 93943</td>
<td>2</td>
</tr>
<tr>
<td>Defense Technical Information Center Cameron Station Alexandria, VA 22314</td>
<td>2</td>
</tr>
<tr>
<td>Department of Operations Research Naval Postgraduate School Monterey, CA 93943</td>
<td>20</td>
</tr>
<tr>
<td>ATTN: Prof. Robert R. Read, Code OR-Re</td>
<td></td>
</tr>
<tr>
<td>Department of Operations Research Naval Postgraduate School Monterey, CA 93943</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: Prof. R. N. Forrest, Code OR-Fo</td>
<td></td>
</tr>
<tr>
<td>Naval Postgraduate School Monterey, CA 93943</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: Research Administration, Code 012</td>
<td></td>
</tr>
<tr>
<td>Department of Operations Research Naval Postgraduate School Monterey, CA 93943</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: Prof. P. Purdue, Code OR-Pd</td>
<td></td>
</tr>
<tr>
<td>Department of Operations Research Naval Postgraduate School Monterey, CA 93943</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: Prof. J. Esary, Code OR-Ey</td>
<td></td>
</tr>
<tr>
<td>Department of Operations Research Naval Postgraduate School Monterey, CA 93943</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: Prof. O. B. Wilson, Code PH-WI</td>
<td></td>
</tr>
</tbody>
</table>
Department of Operations Research
Naval Postgraduate School
Monterey, CA  93943
ATTN:  Prof. L. Johnson, Code OR-Jo

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  R. Mash, Code 50

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  R. D. Helander, Code 51

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  S. L. McKeel, Code 512

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  J. Knudsen, Code 5122

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  T. Ward, Code 5122

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  G. Olsen, Code 5122

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  J. Hall, Code 5121

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  R. Evans, Code 5113

Naval Undersea Warfare Engineering Station
Keyport, Washington  98345
ATTN:  L. Beer, Code 5113