Locating microseismic events and traveltime mapping using locally spherical wavefronts

John C. Bancroft and Xiang Du

ABSTRACT

A method is presented for identifying the source of a locally circular or spherical wavefront given the traveltimes at arbitrary locations. For 2D data, the center of the circular wavefront is computed from three traveltimes recorded at three arbitrary locations. Application to 3D data requires four traveltimes recorded at four arbitrary locations.

This method is suited for a number of applications such as mapping traveltimes that are computed along sparse raypaths to gridded traveltimes, the monitoring of micro-seismic events caused by fraccing, or to the possible prediction of landslides in geologically unstable areas.

The analytic solution for the 2D problem is found using the tangency of circles, a problem that was originally solved about 200 BC by Apollonius, a Greek mathematician. The 3D solution involves the tangency of spheres and was obtained by using a parallel development to the 2D solution. Both the 2D and 3D problems produce two solutions from which one must be chosen.

INTRODUCTION

Traveltime computations continue to be an integral part of modelling and seismic imaging by providing efficient kinematic information on the location of propagated energy. The traveltimes may be computed analytically using simplifying assumptions such as RMS velocities, or may be estimated within a complex geological structure using raytracing or gridded traveltimes, as illustrated in Figure 1. Traveltimes on a grid may be computed using a finite difference solution to the Eikonal equation; however that solution is based on a plane wave assumption (Bancroft 2005a).

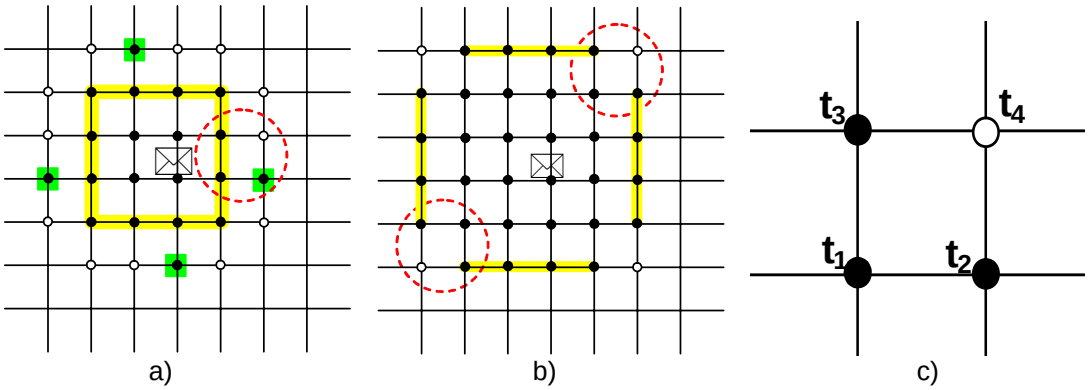


Figure 1. Computing traveltimes on a square grid with a) illustrating the estimation of one point on a side that is offset from the minimum time. The remaining points on the side and the corner points in b), i.e. t_4 are computed from the three known points t_1, t_2, and t_3 as illustrated in part c).

The wavefront is normally curved and in a localised area the curvature can be approximated by a portion circle. Solving for an unknown traveltime, such as the fourth corner of the square, may involve estimating the center of curvature and its source time, then computing the traveltime to the desired location as illustrated in Figure 2. The traveltime computations assume the velocity to be locally constant within the square, however this velocity is extended outside the square to the location of the center of curvature. The solutions usually involve the difference in the known traveltimes and the intersection of corresponding hyperbolae.

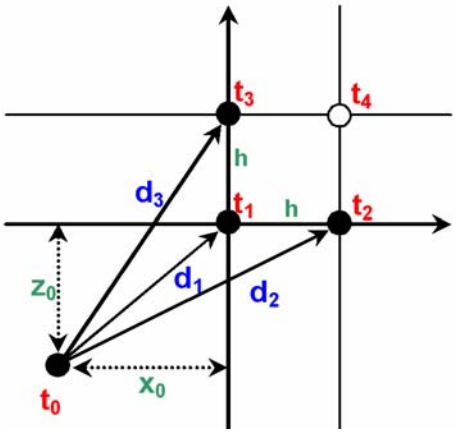
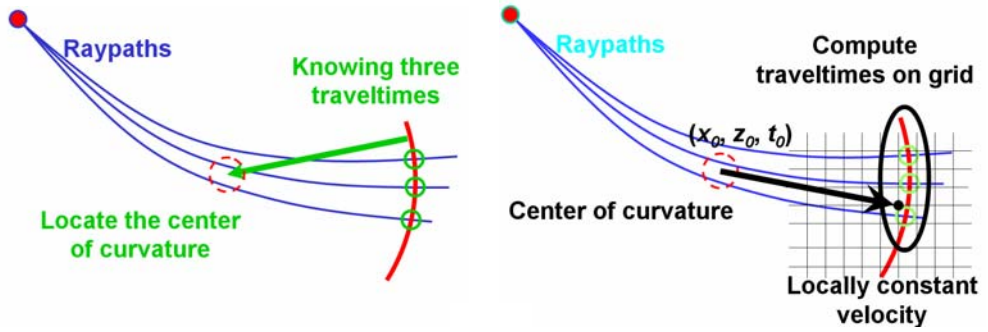


Figure 2. Illustration of the geometry for the circular wavefront assumption where t_1, t_2, and t_3 are the known traveltimes, t_4 the unknown traveltime. The apparent source for these times is (x_0, z_0, and t_0).

Traveltimes are also computed using ray tracing, then the traveltimes along a bundle of rays are used to compute the traveltimes on a grid as in Figure 3.



a) b)

Figure 3. Traveltimes on a bundle of ray in a) are used to compute the center of curvature, which is then used in b) to define the traveltimes on a grid.

Micro-seismic events occur during well fraccing, well injections such as CO_2 or disposal, or in areas of geological stress. Knowledge of the locations of these events, as illustrated in Figure 4, aids in identifying the extent of the fraccing or injection.

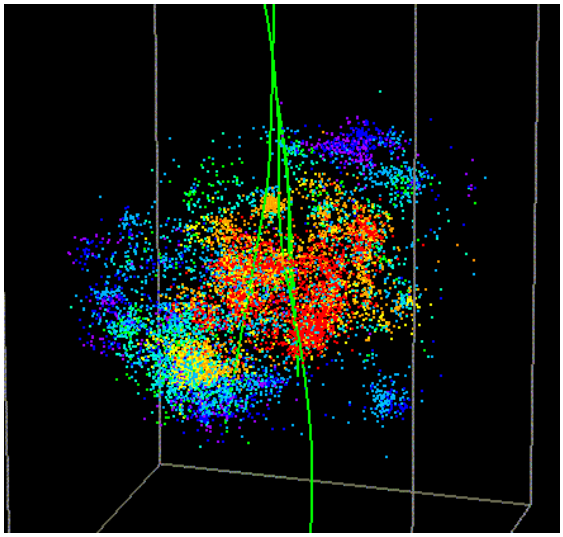


Figure 4. Microseismic events located during a well fraccing process.

Micro-seismic events may also be used to indicate a potential hazard such as an impending landslide or earthquake. These areas of interest are continuously monitored by a number of stationary receivers, and when an event occurs, the location is estimated from the corresponding times of the receivers. A number of algorithms are available for computing the source of the event such as pre-computing the traveltimes from all possible source locations, then using a search technique to find the optimum source location.

Figure 5. Turtle mountain in Alberta is continuously monitored in an attempt to provide warning of a potential land slide.

It is possible to locate the center of curvature using the differences between the traveltimes (Bancroft 2005b). The difference between two traveltimes defines two paths of a hyperbola with the receiver locations at the focus points. One path of the hyperbola can be chosen based on the relative amplitudes of the two traveltimes. Hyperbolic curves can then be defined for each pair of recording points. The intersection points of these

hyperbolae identify potential centers of curvature. The recording points can be at any locations off grid, but the computations become simpler when they are located on regular grid points at the corners of a square as illustrated in Figure 2.

The method presented in this paper is useful for all these applications as it computes the center of curvature using three known times at three locations for 2D data, and four known times and locations for 3D data.

THEORY

Defining the problem

A method for computing the center of curvature from recorded traveltimes is based on the tangency of circles for 2D data and spheres for 3D data. We will develop the method for 2D data as it can be visualized and then extend the method to 3D using parallel development.

We start by assuming a source location (x_0, z_0) and three arbitrarily located receiver points (x_1, z_1), (x_2, z_2), and (x_3, z_3) as illustrated in Figure 6a. The traveltimes recorded at these locations are t_1, t_2, and t_3 clock times, i.e. not the travel times from the source point. We assume the event started at the source location at time t_0 which is also a clock time that is not zero.

The delta traveltimes between the source and receiver locations are then defined by $t_{01} = (t_1 - t_0)$, $t_{02} = (t_2 - t_0)$, and $t_{03} = (t_3 - t_0)$. These traveltimes may be used to define radii r_{01}, r_{02}, and r_{03} or distances from the source to the receiver locations with the appropriate constant velocity v as illustrated with circles in Figure 6b.

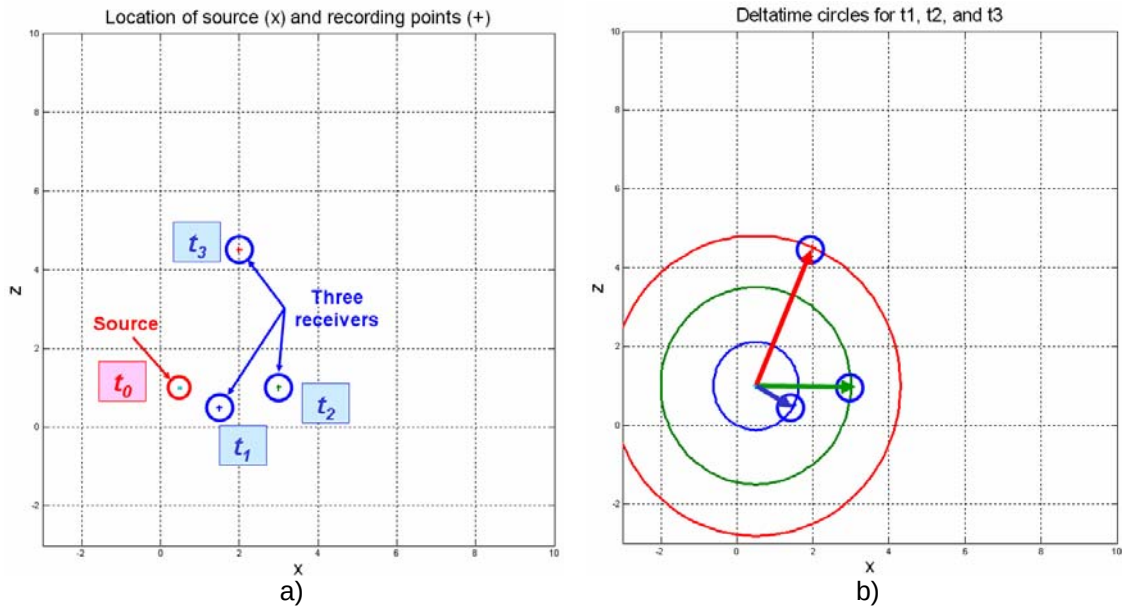


Figure 6. Model used to set up the problem a) of three receivers on a 2-D pane and b) circles drawn with the delta times.

Rather than draw circles centered at the source point, we draw circles centered at the receiver locations as illustrated in Figure 7. The intersection of these circles define the source location. However, we do not know these radii at the beginning of our problem.

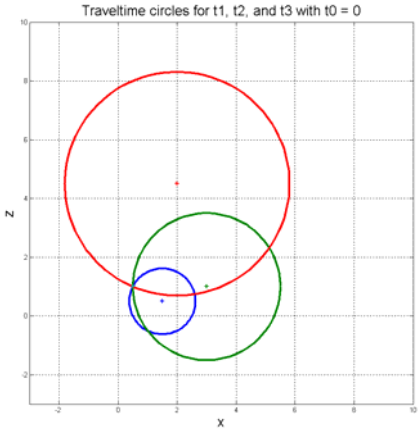
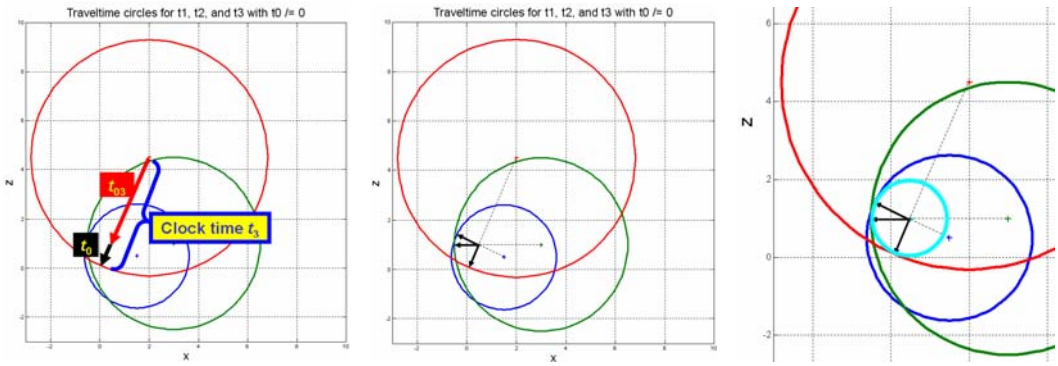


Figure 7. Circles drawn with source-to-receiver radii, centered at the receiver. The intersection of these circles is the source location that we now know, but we will be trying to find later. Notice that the radii are defined using the source time t_0, which is known at this time but which we will be also trying to find.

The solution

Our problem is given the times t_1, t_2, and t_3, we can draw circles representing their radii as illustrated in Figure 8a. The radii of these circles is the same as the sum of the source time t_0 plus the delta time to the receiver, i.e. $r_1 = v(t_0 + t_{01})$, $r_2 = v(t_0 + t_{02})$, and $r_3 = v(t_0 + t_{03})$. Consider the three radii to pass through the source point as in Figure 8b. These three radii pass beyond the source point with the same distance $v*t_0$ as illustrated in Figures 8b and 8c. A fourth circle may now be drawn through the three points defined where the arrowheads meet the circles (Figure 8c). This circle is tangent to the other three circles, is unique, and its center coincides with the source location. *Finding the center and radius of this circle will solve our problem.*



a) b) c)

Figure 8. Circles drawn from the known locations with the clock times t_1, t_2, and t_3 in a) with b) showing arrows representing the distances past the source location that are normal to the circles, and c) the inclusion of a cyan coloured circle that is tangent to the circles at these same points. The center of this circle is the source location and it radius represents the source time t_0.

Solving the 2-D problem

The problem now becomes one of finding a circle that is tangent to three given circles. It has been of interest for centuries with the first solution attributed to Apollonius of Perga who was born in 261 BC. Geometric constructions are possible but complex, illustrating that a solution should "involve nothing more than quadratic equations", (Website 1). Numerous algebraic solutions are also available on the web, and we will follow one based on Website 2, that defined in the list of references.

We start with equations representing the three circles with radii from the source to the three receiver locations, i.e.

$$(x_1 - x_0)^2 + (z_1 - z_0)^2 = v^2(t_1 - t_0)^2 \tag{1}$$

$$(x_2 - x_0)^2 + (z_2 - z_0)^2 = v^2(t_2 - t_0)^2 \tag{2}$$

$$(x_3 - x_0)^2 + (z_3 - z_0)^2 = v^2(t_3 - t_0)^2 . \tag{3}$$

Subtracting equations (2) and (3) from (1) we get

$$\begin{aligned} 2x(x_1 - x_2) + 2z(z_1 - z_2) + (x_2^2 - x_1^2) + (z_2^2 - z_1^2) = \\ 2v^2(t_1 - t_2)t_0 + v^2(t_2^2 - t_1^2) \end{aligned} \tag{4}$$

and

$$\begin{aligned} 2x(x_1 - x_3) + 2z(z_1 - z_3) + (x_3^2 - x_1^2) + (z_3^2 - z_1^2) = \\ 2v^2(t_1 - t_3)t_0 + v^2(t_3^2 - t_1^2) \end{aligned} . \tag{5}$$

These equations are simplified to

$$A_1 x_0 + B_1 z_0 + C_1 t_0 = D_1 \tag{6}$$

$$A_2 x_0 + B_2 z_0 + C_2 t_0 = D_2 , \tag{7}$$

where

$$A_1 = 2(x_1 - x_2), \quad B_1 = 2(z_1 - z_2), \quad C_1 = -2v^2(t_1 - t_2) \tag{8}$$

$$D_1 = v^2(t_2^2 - t_1^2) - (x_2^2 - x_1^2) - (z_2^2 - z_1^2) \tag{9}$$

and

$$A_2 = 2(x_1 - x_3), \quad B_2 = 2(z_1 - z_3), \quad C_2 = -2v^2(t_1 - t_3) \tag{10}$$

$$D_2 = v^2(t_3^2 - t_1^2) - (x_3^2 - x_1^2) - (z_3^2 - z_1^2). \tag{11}$$

Solving equations (6) and (7) for x_0 and z_0, we get

$$x_0 = \frac{B_2(D_1 - C_1 t_0) - B_1(D_2 - C_2 t_0)}{A_1 B_2 - A_2 B_1} = E_1 t_0 + F_1 \tag{12}$$

$$z_0 = \frac{A_2(D_1 - C_1 t_0) - A_1(D_2 - C_2 t_0)}{A_2 B_1 - A_1 B_2} = E_2 t_0 + F_2, \tag{13}$$

where

$$E_1 = \frac{B_1 C_2 - B_2 C_1}{A_1 B_2 - A_2 B_1}, \quad F_1 = \frac{B_2 D_1 - B_1 D_2}{A_1 B_2 - A_2 B_1} \tag{14}$$

$$E_2 = \frac{A_1 C_2 - A_2 C_1}{A_2 B_1 - A_1 B_2}, \quad F_2 = \frac{A_2 D_1 - A_1 D_2}{A_2 B_1 - A_1 B_2}. \tag{15}$$

Now substitute equations (12) and (13) into equation (1) to get

$$(E_1 t + F_1 - x_1)^2 + (E_2 t_0 + F_2 - z_1)^2 = v^2(t_0 - t_1)^2, \tag{16}$$

where t_0 is the only unknown. Rewriting this equation in quadratic form for t_0 we get

$$\begin{aligned} &t_0^2(v^2 - E_1^2 - E_2^2) + \\ &t_0[-2v^2 t_1 + 2(x_1 - F_1)E_1 + 2(z_1 - F_2)E_2] + . \\ &v^2 t_1^2 - (F_1 - x_1)^2 - (F_2 - z_1)^2 = 0 \end{aligned} \tag{17}$$

This equation simplifies to

$$t_0^2 + G t_0 + H = 0, \tag{18}$$

with solutions

$$t_0 = \frac{-G \pm \sqrt{G^2 - 4H}}{2}, \tag{19}$$

where

$$G = \frac{-2v^2 t_1 + 2(x_1 - F_1)E_1 + 2(z_1 - F_2)E_2}{v^2 - E_1^2 - E_2^2} \tag{20}$$

and

$$H = \frac{v^2 t_1^2 - (F_1 - x_1)^2 - (F_2 - z_1)^2}{v^2 - E_1^2 - E_2^2}. \tag{21}$$

Once we have t_0 from equation (19) we then use equations (12) and (13) to solve for x_0 and z_0. MATLAB code for this solution is included in the appendix as TT2D.m.

2-D EXAMPLES

To test this algorithm we show two cases where we start by defining the source and receiver locations. We then define a value for t_0, and compute t_1, t_2, and t_3. Using only the receiver times and their locations, we compute the two solutions for x_0, z_0, and t_0. For the two examples given, Solution 1 uses the plus sign in equation (19) while Solution 2 uses the minus sign solution. The geometry for the first example is illustrated in Figures 6 through 8. Figure 9 shows the geometry of the alternate solution. Note in the alternate solution for t_0 the circle is tangent to the outside of the circles. The input and results for both cases are included in the appendix.

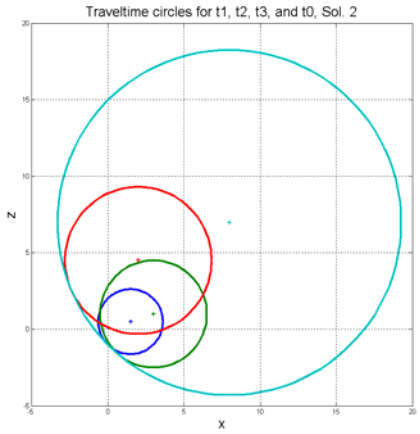


Figure 9. Solution 2 for Example 1 where the t_0 circles is exterior to the three known circles.

Example 2 is illustrated below in Figure 10. Receiver x_2 is shifted to the right, enough to change the correct solution to Solution 2 that defines the inner and desired solution.

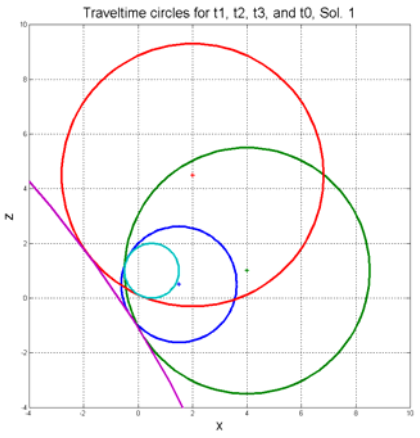


Figure 10. Two solutions for Example 2; the first solution is illustrated with the purple curve that is tangent to the outside of the three circles, while the second desired solution is illustrated with a cyan circle.

Solution for 3D data

The 3D solution has four unknowns, x_0, y_0, z_0 and t_0. and we therefore require four independent equations, or four times at four receiver locations. Similar to the 2D case, we start with equations that define spheres from the receiver locations that pass through the source location.

$$(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 = v^2(t_1 - t_0)^2 \tag{22}$$

$$(x_2 - x_0)^2 + (y_2 - y_0)^2 + (z_2 - z_0)^2 = v^2(t_2 - t_0)^2 \tag{23}$$

$$(x_3 - x_0)^2 + (y_3 - y_0)^2 + (z_3 - z_0)^2 = v^2(t_3 - t_0)^2 \tag{24}$$

$$(x_4 - x_0)^2 + (y_4 - y_0)^2 + (z_4 - z_0)^2 = v^2(t_4 - t_0)^2. \tag{25}$$

Using the same procedure outlined for the 2D case, and tedious algebra we end up with a solution similar to equation (19) that has two possible solutions for t_0, and similar equations that compute x_0, y_0, and z_0.

COMMENTS

In the above solutions, the time t_0 is quite arbitrary and can be virtually any value. The radii of the circles in Figure 3 are dependent on this value and can be very large if t_0 is very large, and could create inaccuracies in the solutions. Since t_0 is arbitrary, so also are the times t_1, t_2, and t_3 in the sense that we could subtract a constant time from each of the values. Subtracting the minimum recorded time from all recorded times will leave one time at zero and the remaining positive, while the source point will have a negative time. This will have the effect of minimizing the radius of the time circles, with one reduced to a point. The solution now becomes one of fitting a circle through one point and tangent to two circles that should be more efficient than fitting a circle to three other circles.

CONCLUSIONS

A method was presented that locates the center of curvature for a curved wavefront when given traveltimes at known receiver locations. The receivers may be arbitrarily located.

Applications are suitable for locating micro-seismic events or for mapping raypath traveltimes to a grid.

REFERENCES

Bancroft, J. C., 2005a, Circular wavefront assumptions for gridded traveltime computations: CREWES Research Report, **17**.

Bancroft, J. C., 2005b, The computation of gridded traveltimes when assuming circular wavefronts: 82nd Ann Internat. Mtg, Soc. Expl. Geophys., Expanded Abstracts

Website 1, htpp://www.mathpages.com/home/kmath113.htm

Website 2, hrpp://mathworld.wolfram.com/ApolloniusProblem.html

ACKNOWLEDGEMENTS

The authors thank the CREWES sponsors for supporting this work.

APPENDIX

MATLAB code for 2-D solution (`TT2D.m`)

The following code defines the location of the three receivers. It also defines the location of a source point and its t_0 time that is to be estimated. The times at the receivers are computed. Using these receiver times and the known locations of the receivers, the location and t_o time of the source is computed. The output of this code was used to create Figures 6 through 10. The solution to the parameters in this routine are shown at the end of this listing.

Different locations of the source and receivers will provide correct answers that may be either the initial or alternate solution.

```
%  TT2D.m  Compute the source location and initial time  from the
%  recording time at three known locations.  All point lie on the
%  same plane.
%  Assumption of this solution.
%  1. Consider a line from a recording location at A (x1, y1) with time t1
%     that passes through the source point at B (x0, y0) with time t0.
%     The traveltime between these two point is t1 - t0.  Extend that line
%     past the source point by the source time t0.  That new location at A
%     represents time zero for that recording point at A (x1, y1).

%              C            B                                  A
%              *............*..................................*
%            t=0           t=t0                               t=t1
%                <---t0-----> <---------- t1 - t0 ------------>
%                 <-----------------t1-----------   ------------>

%              |<--------------------Rad1----------------------0

%              |<----Rad0-->0


%
%  2. Now consider one circle drawn from the recording point at A
%     with a radius of Rad1, and a second circle drawn from B with radius
%     Rad0 as illustrated above.  These two circle meet at the tangent
%     point C.

%  3. Similar constructions could be done for the other two recording
%     locations.  All three circles will be tangent to the same circle
%     from the source point.

%  4. Since the radii Rad1, Rad2, and Rad3 are known, the problem remains
%     to find the source location and it time.  This is accomplished by
%     finding the fourth circle (the source Rad0) that is tangent to the
%     other three circles.

%  5. The problem of drawing a circle that is tangent to three other
%     circle is a well known mathematical problem that was posed many
%     years ago by Apollonius, a Greek mathematician.
```

```
clear all;

% First condition
% x1=1000;z1=0;t1=1;
% x2=-1500;z2=0;t2=1.51;
% x3=0;z3=-1000;t3=1.0;
% v=1000;

% First condition
% v=1000;
% x0 = 100; z0 = -100; t0 = 0.05
% x1=700;z1=700;t1=1.00;
% x2=500;z2=200;t2=0.50;
% x3=-300;z3=200;t3=0.5;

% John's condition
%_____________________________________________________________________
v = 1.0;                %   Velocity
x0 = 0.5;  z0 = 1.0; t0 = 1.0;   %   Theoretical location of unknown
x1 = 1.5;  z1 = 0.5;     %   Three given points
%x2 = 3.761;  z2 = 1.0;
x2 = 3.0;  z2 = 1.0;
x3 = 2.0;  z3 = 4.5;
%
% x0 = -5;  z0 = 0.0; t0 = 1.0;   %   Theoretical location of unknown
% x1 = 2.0;  z1 = 1.0;     %   Three given points
% x2 = 2.0;  z2 = 0.0;
% x3 = 2.0;  z3 = -1.0;

%x0 = -5;  z0 = 0.0; t0 = 2.0;   %   Theoretical location of unknown
%x1 = -1.0;  z1 = 3.0;     %   Three given points
%x2 = 1.0;  z2 = 1.0;
%x3 = 2.0;  z3 = 0.0;

r0 = t0*v;
d0 = r0;
d1 = sqrt(  (x1-x0)^2 + (z1-z0)^2); % Distances from srce to three known points
d2 = sqrt(  (x2-x0)^2 + (z2-z0)^2); % Distances from srce to three known points
d3 = sqrt(  (x3-x0)^2 + (z3-z0)^2); % Distances from srce to three known points

t1 = t0 + d1/v;     % Time at the three known points
t2 = t0 + d2/v;
t3 = t0 + d3/v;
r1 = t1*v;      % For plotting only
r2 = t2*v;
r3 = t3*v;
%_________________________

A1=2*(x1-x2); B1=2*(z1-z2); C1=-2*v*v*(t1-t2);
A2=2*(x1-x3); B2=2*(z1-z3);
C2=-2*v*v*(t1-t3);
D1=v*v*(t2*t2-t1*t1)-(x2*x2-x1*x1)-(z2*z2-z1*z1);
D2=v*v*(t3*t3-t1*t1)-(x3*x3-x1*x1)-(z3*z3-z1*z1);

E1=(B1*C2-B2*C1)/(A1*B2-A2*B1);  F1=(B2*D1-B1*D2)/(A1*B2-A2*B1);
E2=(A1*C2-A2*C1)/(A2*B1-A1*B2);  F2=(A2*D1-A1*D2)/(A2*B1-A1*B2);

G=(-2*v*v*t1+2*(x1-F1)*E1+2*(z1-F2)*E2)/(v*v-E1*E1-E2*E2);
H=(v*v*t1*t1-(F1-x1)*(F1-x1)-(F2-z1)*(F2-z1))/(v*v-E1*E1-E2*E2);

% _____________________% Solution 1_____________________
t01=(-G-sqrt((G*G-4*H)))/2.0;
x01=(B2*(D1-C1*t01)-B1*(D2-C2*t01))/(A1*B2-A2*B1);
z01=(A2*(D1-C1*t01)-A1*(D2-C2*t01))/(A2*B1-A1*B2);
```

```
r01=v*t01;
t11 = t01 + sqrt( (x1-x01)^2 + (z1-z01)^2 )/v;
t21 = t01 + sqrt( (x2-x01)^2 + (z2-z01)^2 )/v;
t31 = t01 + sqrt( (x3-x01)^2 + (z3-z01)^2 )/v;
% _____________________% Solution 1_____________________

% _____________________% Solution 2_____________________
t02=(-G+sqrt((G*G-4*H)))/2.0;
x02=(B2*(D1-C1*t02)-B1*(D2-C2*t02))/(A1*B2-A2*B1);
z02=(A2*(D1-C1*t02)-A1*(D2-C2*t02))/(A2*B1-A1*B2);

r02=v*t02;
t12 = t02 + sqrt( (x1-x02)^2 + (z1-z02)^2 )/v;
t22 = t02 + sqrt( (x2-x02)^2 + (z2-z02)^2 )/v;
t32 = t02 + sqrt( (x3-x02)^2 + (z3-z02)^2 )/v;
% _____________________% Solution 2_____________________

disp(['The defined valuess are ']);
disp(['t0 =' num2str(t0 )  '   x0 =' num2str(x0 )  '   z0 =' num2str(z0 ) ]);
disp(['t1 =' num2str(t1) '    t2 =' num2str(t2 )  '   t3 =' num2str(t3 )   ]);
disp([' ']);
disp(['The Solution 1 results are ']);
disp(['t01=' num2str(t01)  '   x01=' num2str(x01)  '   z01=' num2str(z01) ]);
disp(['t11=' num2str(t11) '    t21=' num2str(t21)  '   t31=' num2str(t31)   ]);
disp([' ']);
disp(['The Solution 2 results are ']);
disp(['t02=' num2str(t02)  '   x02=' num2str(x02)  '   z02=' num2str(z02) ]);
disp(['t12=' num2str(t12) '    t22=' num2str(t22)  '   t32=' num2str(t32)   ]);

t = 0:pi/50:2*pi;       % For parametric plot

figure(1); plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x0,z0,'x', 'LineWidth',2 );
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Location of source (x) and recording points (+)'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-3 10 -3 10]);

figure(2); plot(x1+d1*cos(t),z1+d1*sin(t), x2+d2*cos(t),z2+d2*sin(t),
x3+d3*cos(t),z3+d3*sin(t), 'LineWidth',3 );
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x0,z0,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Traveltime circles for t1, t2, and t3 with t0 = 0'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-3 10 -3 10]);

figure(3); plot(x1+r1*cos(t),z1+r1*sin(t), x2+r2*cos(t),z2+r2*sin(t),
x3+r3*cos(t),z3+r3*sin(t), 'LineWidth',3);
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x01, z01,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Traveltime circles for t1, t2, and t3 with t0 /= 0'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-3 10 -3 10]);

figure(4); plot(x1+r1*cos(t),z1+r1*sin(t), x2+r2*cos(t),z2+r2*sin(t),
x3+r3*cos(t),z3+r3*sin(t), x01+r01*cos(t),z01+r01*sin(t), 'LineWidth',3 );
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x01, z01,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Traveltime circles for t1, t2, t3, and t0, Sol. 1'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-4 10 -4 10]);
```

```
figure(5); plot(x1+r1*cos(t),z1+r1*sin(t), x2+r2*cos(t),z2+r2*sin(t),
x3+r3*cos(t),z3+r3*sin(t), x02+r02*cos(t),z02+r02*sin(t), 'LineWidth',3);
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x02,z02,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Traveltime circles for t1, t2, t3, and t0, Sol. 2'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-4 10 -4 10]);

figure(6); plot(x1+r1*cos(t),z1+r1*sin(t), x2+r2*cos(t),z2+r2*sin(t),
x3+r3*cos(t),z3+r3*sin(t), x01+r01*cos(t),z01+r01*sin(t),
x02+r02*cos(t),z02+r02*sin(t),   'LineWidth',3 );
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x01,z01,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Traveltime circles for t1, t2, t3, and t0, Sol. 1'];
title(ttl,'FontSize',20);
grid on; axis equal; axis([-4 10 -4 10]);

figure(7); plot(x0+d1*cos(t),z0+d1*sin(t), x0+d2*cos(t),z0+d2*sin(t),
x0+d3*cos(t),z0+d3*sin(t), 'LineWidth',3 );
hold on; plot(x1,z1,'+',x2,z2,'+',x3,z3,'+',x0,z0,'+', 'LineWidth',2 ); hold
off
xlabel('x','FontSize',20), ylabel('z','FontSize',20)
ttl = ['Deltatime circles for t1, t2, and t3']; title(ttl,'FontSize',20);
grid on; axis equal; axis([-3 10 -3 10]);

The defined valuess are
t0 =1   x0 =0.5   z0 =1
t1 =2.118    t2 =3.5   t3 =4.8079

The Solution 1 results are
t01=1   x01=0.5   z01=1
t11=2.118    t21=3.5   t31=4.8079

The Solution 2 results are
t02=11.2754   x02=7.975   z02=6.9754
t12=20.4327    t22=19.0508   t32=17.7429
>>
```

Example Case 1 in text

```
Input
v = 1.0;  % Velocity
x0 = 0.5;  z0 = 1.0; t0 = 1.0;
x1 = 1.5;  z1 = 0.5;
x2 = 3.0;  z2 = 1.0;
x3 = 2.0;  z3 = 4.5;
Output
Defined values t0=1   x0 =0.5   z0 =1
Solution 1     t01=1  x01=0.5   z01=1
Solution 2 t02=11.27 x02=7.975 z02=6.97
```

Example Case 2 in text

```
Input …
x2 = 4.0;  z2 = 1.0;
Output
Defined values t0 =1   x0 =0.5   z0 =1
Solution 1 t01=-41.45 x01=-35.13 z01=-23.0
Solution 2     t02=1   x02=0.5   z02=1
```

MATLAB code for 3D solution (`TT3D.m`)

The following code defines the location of the four receivers. It also defines the location of a source point and its t_o time that is to be estimated. The times at the receivers are computed. Using these receiver times and the known locations of the receivers, the location and t_o time of the source is computed. At this time there is no graphical output because of the difficulty viewing the intersecting spheres. The output is text that displays the defined source location and time and the two estimated source locations and time, one of which will be correct. No attempt has been made to identify which solution is correct, (apart from comparing them to the known solution). Different locations of the source and receivers will provide correct answers that may be either the initial or alternate solution.

```
%   TT3D Compute a 3D location and initial time from four locations
%   by Xiang Du 02/13/06 with the circular wavefront assumption

%   This routine only shows computed values as the spheres can't be
%   plotted as the 2D case with circles on a plane.  (TT2D.m)

%   In microseismic work, it is recommended to subtract the smallest
%   time from all recorded times to keep the circles as small as
%   possible. In this case to will be negative.

clear all;

% vvvvvvvvvvvvvvvv Set up initial test condition vvvvvvvvvvvvvvvvvvv
v = 2.0;              %   Velocity
x0 = 0.5;  y0 = -0.5; z0 = 1.0; t0 = 2.0; %Location of unknown

x1 = 1.5;  y1 =  0.5; z1 = 0.5;    %   Four given receiver points
x2 = 2.5;  y2 =  1.5; z2 = 1.0;
x3 = 2.0;  y3 =  0.3; z3 = 4.0;
x4 = 1.0;  y4 =  1.5; z4 = 2.0;

%   Compute distances to the known source point
d1 = sqrt(  (x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2);     % Distances from
theoretical source to three known points
d2 = sqrt(  (x2-x0)^2 + (y2-y0)^2 + (z2-z0)^2);     % Distances …
d3 = sqrt(  (x3-x0)^2 + (y3-y0)^2 + (z3-z0)^2);     % Distances …
d4 = sqrt(  (x4-x0)^2 + (y4-y0)^2 + (z4-z0)^2);     % Distances …

%   Compute the traveltime to the four known points
t1 = t0 + d1/v;
t2 = t0 + d2/v;
t3 = t0 + d3/v;
t4 = t0 + d4/v;
% ^^^^^^^^^^^^^^^^ End set up ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


%   vvvvvvvvvvvvvvv Solution vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
%   Using only the location and traveltimes at the four known points,
%   estimate the location and time of the source point.
A1=2*(x1-x2); B1=2*(y1-y2); C1=2*(z1-z2); D1=-2*v*v*(t1-t2);
E1=v*v*(t2*t2-t1*t1)-(x2*x2-x1*x1)-(y2*y2-y1*y1)-(z2*z2-z1*z1);
```

```
A2=2*(x1-x3); B2=2*(y1-y3); C2=2*(z1-z3); D2=-2*v*v*(t1-t3);
E2=v*v*(t3*t3-t1*t1)-(x3*x3-x1*x1)-(y3*y3-y1*y1)-(z3*z3-z1*z1);
A3=2*(x1-x4); B3=2*(y1-y4); C3=2*(z1-z4); D3=-2*v*v*(t1-t4);
E3=v*v*(t4*t4-t1*t1)-(x4*x4-x1*x1)-(y4*y4-y1*y1)-(z4*z4-z1*z1);


DETA=A1*B2*C3+A2*B3*C1+A3*B1*C2-A1*B3*C2-A2*B1*C3-A3*B2*C1;
F1=(E1*B2*C3+E2*B3*C1+E3*B1*C2-E1*B3*C2-E2*B1*C3-E3*B2*C1)/DETA;
G1=-(D1*B2*C3+D2*B3*C1+D3*B1*C2-D1*B3*C2-D2*B1*C3-D3*B2*C1)/DETA;

F2=(A1*E2*C3+A2*E3*C1+A3*E1*C2-A1*E3*C2-A2*E1*C3-A3*E2*C1)/DETA;
G2=-(A1*D2*C3+A2*D3*C1+A3*D1*C2-A1*D3*C2-A2*D1*C3-A3*D2*C1)/DETA;

F3=(A1*B2*E3+A2*B3*E1+A3*B1*E2-A1*B3*E2-A2*B1*E3-A3*B2*E1)/DETA;
G3=-(A1*B2*D3+A2*B3*D1+A3*B1*D2-A1*B3*D2-A2*B1*D3-A3*B2*D1)/DETA;


H=(-2*v*v*t1+2*(x1-F1)*G1+2*(y1-F2)*G2+2*(z1-F3)*G3)/(v*v-G1*G1-G2*G2-
G3*G3);
I=(v*v*t1*t1-(F1-x1)*(F1-x1)-(F2-y1)*(F2-y1)-(F3-z1)*(F3-z1))/(v*v-
G1*G1-G2*G2-G3*G3);

%vvvvvvvvvvvvvvvvvvvvvvvvvvvvv First solution vvvvvvvvvvvvvvvv
t01=(-H-sqrt((H*H-4*I)))/2.0;
x01=F1+G1*t01;
y01=F2+G2*t01;
z01=F3+G3*t01;
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

% vvvvvvvvvvvvvvv Alternat solution vvvvvvvvvvvvvvvvvvvvv
t02=(-H+sqrt((H*H-4*I)))/2.0;
x02=F1+G1*t02;  % True
y02=F2+G2*t02;  % True
z02=F3+G3*t02;  % True
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


disp([' ']);
disp([' ']);
disp(['Defined values      t0 =' num2str(t01) '   x0 =' num2str(x01) '
y0 =' num2str(y01) '   z0 =' num2str(z01) ]);
disp(['Estimated values    t01=' num2str(t01) '   x01=' num2str(x01) '
y01=' num2str(y01) '   z01=' num2str(z01) ]);
disp(['Alternate solution  t02=' num2str(t02) '   x02=' num2str(x02) '
y02=' num2str(y02) '   z02=' num2str(z02) ]);
```

Output from the above routine

```
Defined values      t0 =2   x0 =0.5  y0 =-0.5   z0 =1
Estimated values    t01=2   x01=0.5  y01=-0.5   z01=1
Alternate solution  t02=6.7109   x02=6.9062  y02=3.3358   z02=5.5485
>>
```