**ELECTRONOTES**
1016 Hanshaw Road
Ithaca, NY 14850

**APPLICATION NOTE NO. 388**

November 1, 2012

# FUN WITH SURVEY MAPS

## INTRODUCTION:

Maps are fun. There are road maps, topographic maps, and online aerial and satellite maps. Today the answer to what you might have found if you had gone down a certain alternative road is available not just from paper maps, but from the computer, virtually instantaneously and in detail. All these are available for various practical purposes as well as for entertainment. And there are survey maps for property.

If you are a property owner, you likely have a deed and corresponding survey map, and may have seen as many as a dozen others. To the ordinary person, these maps are difficult to interpret – except in general. An engineer likely has much more specific notions about the details of the map, including what must be involved in producing one and verifying its accuracy. Yet both the ordinary property owner and the engineer will need to give, at least initially, the same answer to the question "is the map correct?" You don't know. The ordinary person will be thinking of a fee of at least $500 to ask a licensed surveyor if it is correct. The engineer is probably thinking: "This is just a vector summation – I should be able to do it myself." It would seem to the engineer that you just have to figure out the notational conventions.

So you as an engineer know the basic principles. You start somewhere. Usually this is called the "Point of Beginning" (POB). If you are lucky, the start is a pipe or piece of reinforcing rod you are familiar with, or can find. You then find out where "North" is and face in a specified direction relative to North. Proceed in that specific direction a specific distance to a new point. If luck holds, there is another marker there. Keep going around the boundary, locating all pins, and arrive back at the POB. That's the idea, and no engineer need to be told this – it's obvious. Clearly however, even this may perplex a non-engineer. And as for the engineer – actually doing this is not necessarily trivial.

This sort of situation is not unfamiliar to engineers. Many times we understand instructions and/or conceive the pieces of a design or solution quite rapidly, and then the details bring us to the real work.

## WHAT DO WE MEAN BY CORRECTNESS OF A SURVEY?

You are trying to verify that the survey is "correct". The notion of correctness may mean different things, and we never expect a survey to be exact. There will always be measurement errors, finite precision, errors in projecting to a flat surface, etc. But we all understand the notion of being good enough. A survey placing your neighbor's boundary through the center of your house probably fails this test! A survey that says a boundary pin is in one location, and the actual physical pin is 10 feet away is also a serious discrepancy. After all: Which is right? These real physical errors – as opposed to purely mathematical errors – are individually interesting, and probably require individual explanations.

Here however we have in mind a situation where the numerical description in itself just doesn't work. You follow all the angles and distances, and you don't get back to the POB. In most cases, you should get back to within 1/10 of a foot or less (2 cm + 50 pp million). The age of the survey matters. The more recent the survey, the more precision we expect. For very large parcels (segments of perhaps a mile or so) GPS-derived locations and errors of a few feet are likely good enough. For house lots of perhaps an acre or so, recent surveys are likely done with laser ranging, and we expect very precise measurements. Surveys that are 50 years old were likely done with transits and tapes ("chains"), and we may expect errors on the paper of a couple of feet, and the physical corner markers may be rusted out or lost. All these suggest what we expect.

But the map itself, the paper, should not really deteriorate significantly.

## THE LEGAL DESCRIPTION AND THE SURVEY MAP

The "Legal Description" and the "Survey Map" should describe, equivalently, the same thing exactly. If they don't, the legal description is assumed valid over the map. This is silly because the legal description is derived from the map. So let's assume they both are the same. Keep in mind that you are in general trying to verify that a boundary, based on three or more segments (often at least four) closes, based on the corresponding table of angles and distances. Distances are easy to understand. Something like 320.4 feet means just that. We do want our distances in decimal notation, so in the rare event that something might be expressed as feet and inches, convert to decimal.

Angles are a different matter. We really want the angles described in decimal notation, almost certainly degrees. For example, we might well take North to be 0

degrees, and measure degrees clockwise. For example, 97.321 degrees should be just slightly south of due east, and so on. Unfortunately, you are likely to find angles in degrees, minutes, and seconds of arc, and also commonly with reference to dues South. No big problem. A minute is 1/60 of a degree, and a second is 1/60 of a minute of 1/3600 of a degree. So convert. The other thing is that the angles are generally specified with magnitudes of at most 90 degrees. This is why the reference to South is used as well as North. You look North (or South) and then to the East (or West) as specified. It is strongly suggested that instead of working from the description, or the map directly, that you form a tabulation and a sketch as a means of first seeing if your values seem basically right, and for forming the input data to your program. If you have a fairly good map, you can also use a 360° transparent protractor to see if you have the angles figured out reasonably well.

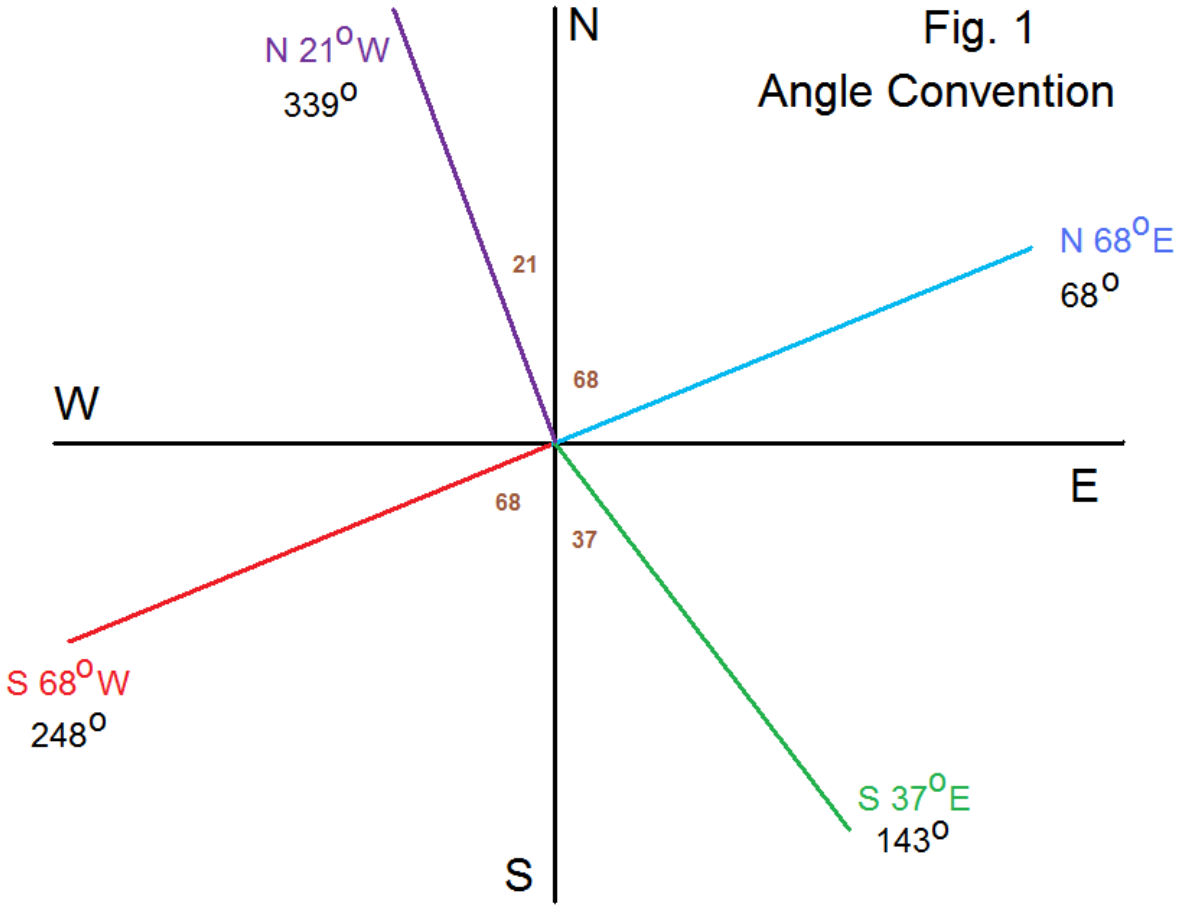



Fig. 1
Angle Convention

Fig. 1 shows the details of the usual angle convention. Keep in mind that we want to convert the convention (N or S heading, E or W tip, degrees, minutes, seconds) to a decimal 360° standard. The figure show four angles, one for each quadrant. Once we determine the quadrant based on N or S, E or W, we then need to add or subtract the magnitude of the angle from 0°, 180°, or 360° as appropriate. Note that the blue and the red angles are on the same line – they just point in opposite directions. It of course makes a difference. If you have only one survey, you can perhaps do these angle conversions by hand.

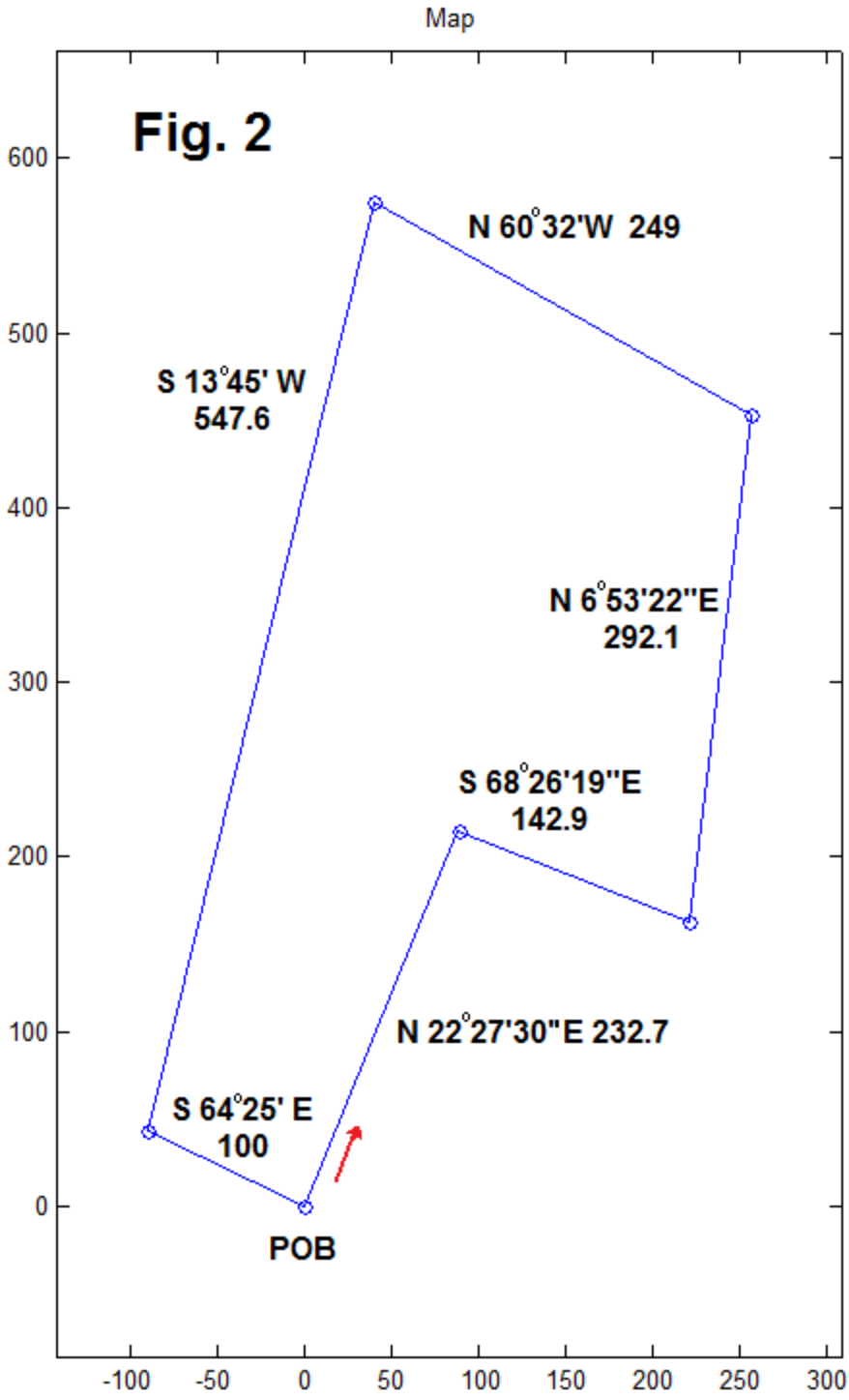
The Matlab program ***survey.m*** was written to more or less input data from a legal description or tabulation. One has only to enter the number of sides and you are prompted for the angles and lengths as (N or S), degrees, minutes, seconds, (E or W), and length. Once all are entered, the table is printed and the map drawn.

## **PROGRAM** *survey.m*

```
clear all
a=[];l=[];

ts=input('Total Sides: ')
% Input/Translate Surveyor's Terminology
for k=1:ts
  side=k;
  display(side);
  NorS=input('N or S: ','s')
  if NorS=='n'; NorS='N';end
  if NorS=='s'; NorS='S';end
  a(k)=input('Angle Magnitude -degrees: ');
  minutes=input('---minutes:');
  sec=input('---seconds:');
  a(k)=a(k)+minutes/60+sec/3600;
  EorW=input('E or W: ','s')
  if EorW=='e'; EorW='E';end
  if EorW=='w'; EorW='W';end
  l(k)=input('Length: ');
  if ((NorS=='S') & (EorW=='E'));a(k)=180-a(k); end
  if ((NorS=='S') & (EorW=='W'));a(k)=a(k)+180; end
  if ((NorS=='N') & (EorW=='W'));a(k)=-a(k); end
end
```

```
a              % angles
l              % lengths

ns(1)=0;ew(1)=0;
for k = 2:length(a)+1
    ns(k)=ns(k-1)+l(k-1)*cos( (a(k-1)/360)*2*pi);
    ew(k)=ew(k-1)+l(k-1)*sin( (a(k-1)/360)*2*pi);
end
  ns             % “y” coordinates
  ew             % “x” coordinates
figure(1)
plot(ew,ns)
hold on
plot(ew,ns,'o')
hold off
ewmax=max(ew);
ewmin=min(ew);
nsmax=max(ns);
nsmin=min(ns);
mew=abs(0.15*(ewmax-ewmin));
mns=abs(0.15*(nsmax-nsmin));
axis('equal');
axis([ewmin-mew,ewmax+mew,nsmin-mns,nsmax+mns]);

title('Map')


complexmap=ew+j*ns;
closureerror=abs(complexmap(length(a)+1))
m=2*closureerror;
figure(2)
plot(0,0,'o')
hold on
plot(ew(k),ns(k),'*')
hold off
axis([-m  m -m  m]);
title('Closure Error o=start *=end')

figure(1)
```

The program calculates and plots as a second figure the closure error. This is really the answer we want. It is either very small, or there is some significant problem. Of course you will also want to compare your Matlab map to your survey map. It is easy to make input errors. Fig. 2 shows an example output map.



The data table printed out for this example is here:

a = 22.4583 111.5614 6.8894 -60.5333 193.7500 115.5833

l = 232.7000 142.9000 292.1000 249.0000 547.6000 100.0000

closureerror = 0.1113

We note that the whole survey closes to within 0.1113 feet. This should be good enough.

## A FEW MORE COMMENTS / FEATURES

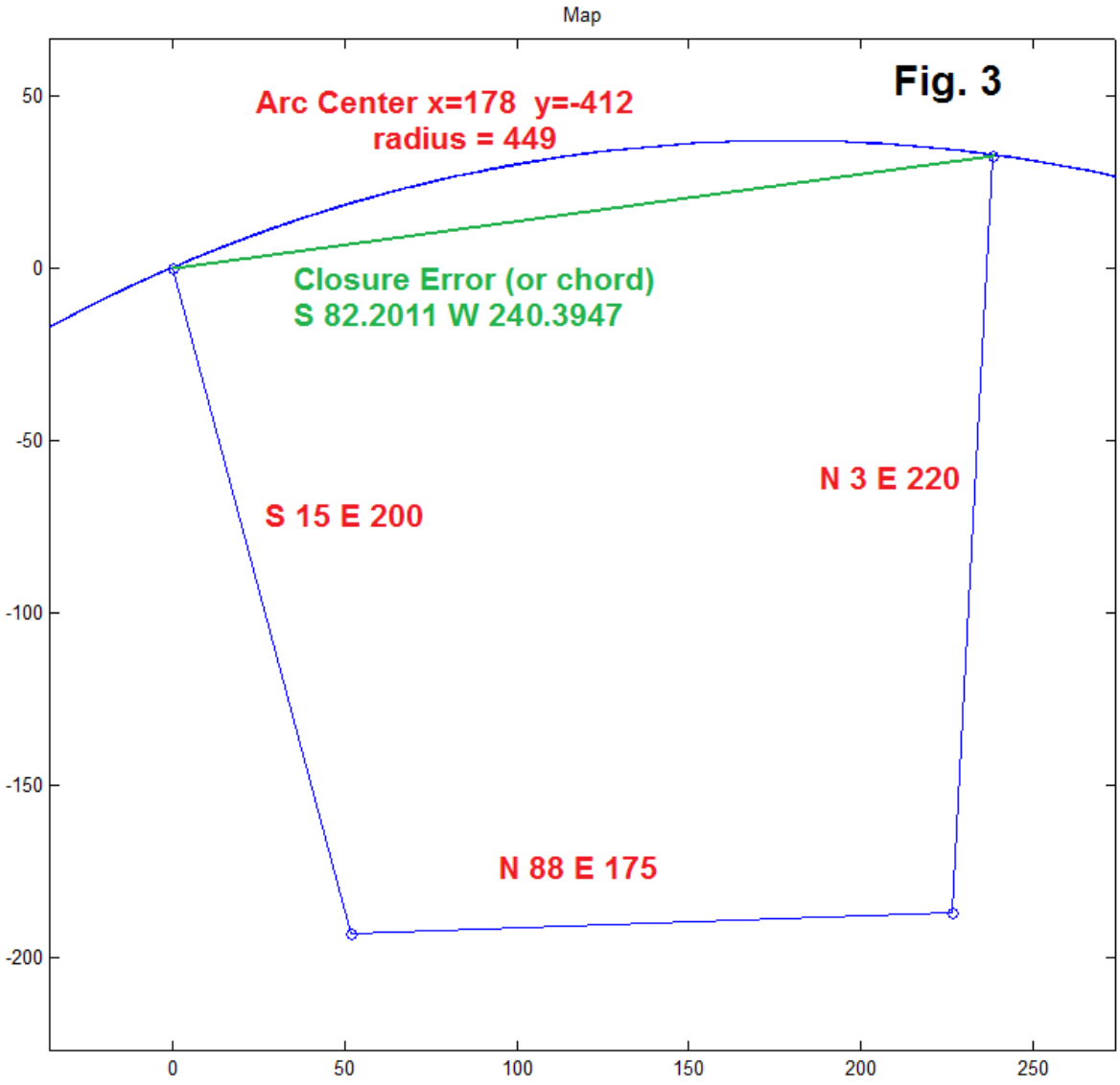
The program here and the resulting example of Fig. 2 deal only with the numerical aspects of the validity of the survey. The survey on paper thus appears okay. Finding out if this is accompanied by a corresponding validity of the physical situation on the ground is a separate issue. In fact, for this particular case, the map does not correspond to the ground – there is a significant error, but you are unlikely to know this if I didn't tell you. Problems like this are likely very common.

You might notice that the first three segments have a precision greater than the last three. It is probably a good idea to find out why such a situation comes up. Here it seems to be due to the fact that the data used here are less the result of actual measurements, and more due to deriving numbers from surrounding surveys. Another situation where the precision "jumps" for one segment is where the segment is computed to be exact. It is obvious that we can use the same sort of computations here, in reverse, to find out what value would give an exact closure. The same precision should probably be used for all segment. When it's not found, some questions may be appropriate.

The coordinate system here is polar (angle and radius), not Cartesian (a rectangular grid). Here we do convert to rectangular (relative to North as y-axis) when it comes to plotting. Sometimes a rectangular grid off some "baseline" is used. One should perhaps ask why if this is found, and request the calibration of any differing reference frames.

Here we deal with straight line boundaries. That's about all you see except for curves that are defined (made to fit) a circle; in which case the center of the circle and the radius are given. Our program does not accommodate this, but it would not be that hard to add. We could just plot the straight lines on either side of the curve, plot the circle, and erase the part we don't need.

The illustration of Fig. 3 makes some of the points above. This is entirely made up, and note that three of the sides have integer values for the angles and for the lengths.




The fourth side is shown in two forms. First there is the green line. This was obtained using the ***survey*** program and inputting only the three lower sides. The fourth side then appears as a closure error – quite substantial, but the math is the same. The program also outputs the corners (o) points, so the angle of the green line can be computed using an inverse tangent. This, and the closure error are shown to four decimal places. The angle is 82.2011° which is 82°12'40". If we put this angle and distance back into the ***survey*** program, the closure error is only 0.0002 feet (or course). The green line looks computed, and of course it was.

Fig. 3 also illustrates the plotting of a curve. We understand straight lines as being useful to a survey. If a line is not straight, probably because it is not straight on the ground (like frontage along a curving roadway) it is probably represented as a curve that is going to be fit to a circle. There are two points to fit this curve to (the ends of the straight segments) so if we had a third point, we understand that we can compute the center (two numbers) and radius (a third number) to fit. We have not done this here (but see the Appendix below), but merely shown a possible curve. In this instance, the "closure error" is a chord of the circle. Remember that the boundary line here is a curve because the surveyor maps it that way. Perhaps you want to argue that some other boundary shape is the actual case – but we are after a good-enough approximation after all.

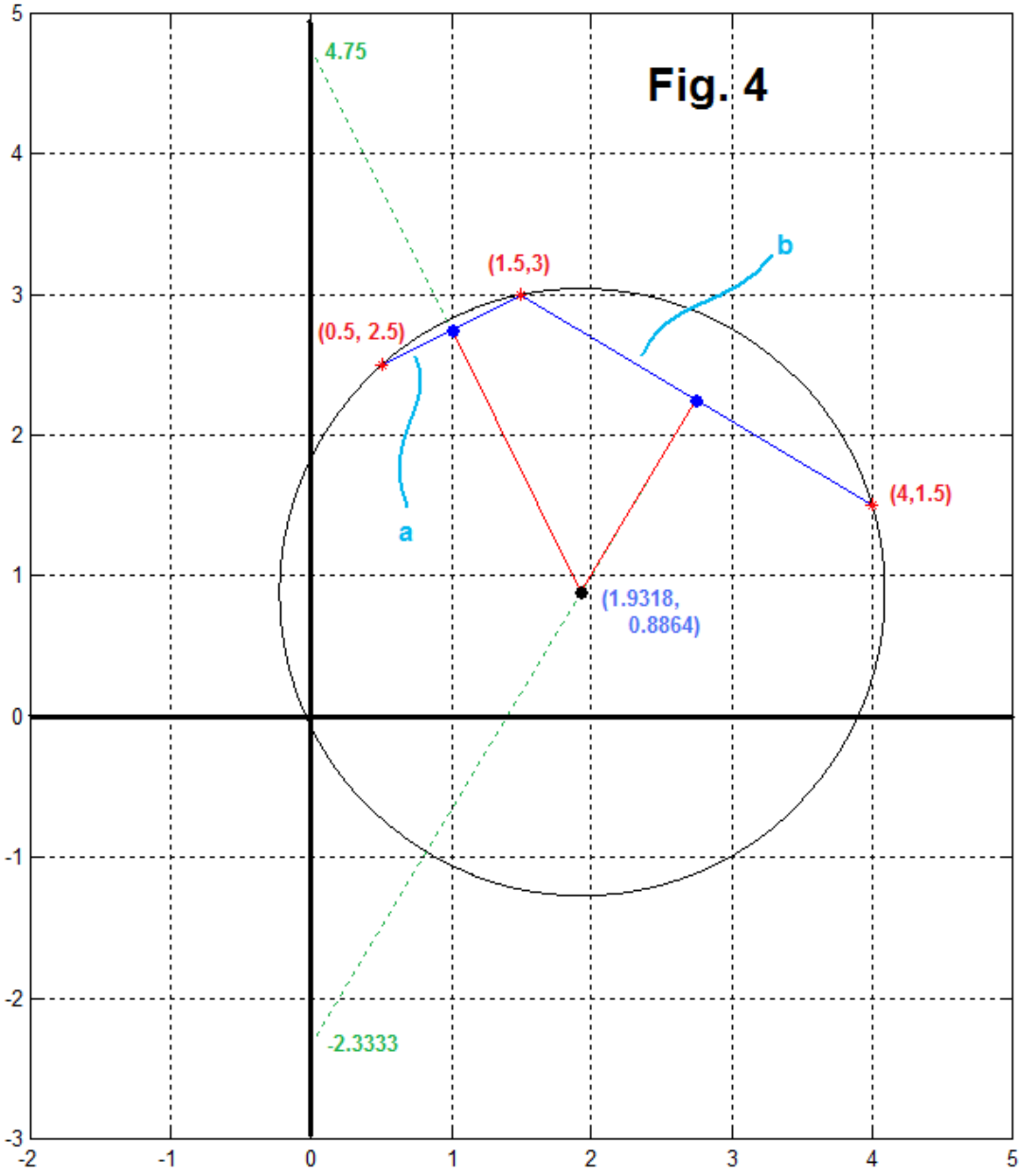
One other case where there may be a non-straight side is where, as is often the case, the boundary is a stream of some sort. The boundary may be defined in the deed to be the stream, and this could well wiggle about – certainly not be straight or even a circle. In such a case, the straight line between pins (or computed points in the streambed) where the boundary leaves the stream can be plotted (like the chord of the arc).

* * * * * * * * * * *

There is probably enough here for anyone checking an individual survey to work with and avoid a problem or two that might come up without some guidance, despite the fact that you suspect that you really do know enough to do everything yourself. One is tempted however to suppose that one might write a program that inputs the text of the legal description, parses it for the angles and lengths, and computes the map – perhaps initially just reporting if it appears to close sufficiently, or not. In this case, it is probably clear that the mathematics is the easy part. There are in just about every description a few things (like "passing through a pipe at 281.1 feet") to break up any supposed order or possible tabulation.

## APPENDIX – FITTING A CIRCLE TO THREE POINTS

This is a standard exercise that is first viewed as a geometry construction (compass and straight edge) and then we can write down equations and computer code. The idea is that the perpendicular bisector of a chord of a circle passes through the center. Thus if we have two non parallel chords, the two perpendicular bisectors intersect at the center. A glance at Fig. 4 is possibly all most readers need to be reminded of this method. This is likely the way the curves were fit in older surveys.




In the figure, we show three points (red stars), which we assume are not originally derived from any particular circle. That is, you don't have the circle there – yet. We draw "chord lines" between the two pairs of points (blue lines). We do not have to use these pairs – for example the first and second and then the first and third would also work. Using a compass, we would construct the perpendicular bisectors (red lines) and where they intersect (black point) we have the center of the circle. Then we would place the point of the compass at the center, and the pen on any of the three original points (it doesn't matter) and draw the circle. You probably have a good idea of the general techniques required to code up this procedure on a computer.

The program code ***circlefit.m*** is printed below. The first step is to input the x and y coordinates of the three points to fit. Avoid placing points that could cause divide-by-zero errors. This could occur with either vertical or horizontal slopes, so offset your inputs ever so slightly if you have a horizontal or vertical allignment to start with.

The first computation is to find the midpoints of the chords between the designated points (blue lines). This is trivial, just the averages of the coordinates at the endpoints of the chord. Then you need the slopes of the chords. Again this is trivial – just subtracting coordinates and dividing the y difference by the x difference. Now – what is the slope of a line that is perpendicular to another line? Recall (or derive) that the slope of a line perpendicular to one that has slope m is -1/m. The slopes of the chords are 1/2 and -3/5, so the slopes of the bisectors are -2 and 1.6667 for our example.

So now we know something about the perpendicular bisectors. We know their slopes, and we know one point on the lines (the midpoints of the chords). That's enough. The general formula for a line is y = mx + b where m is the slope and b is the "y–intercept". We know the m of the perpendicular, and we have an x and a y value for the midpoint on the line. So we compute the intercepts, solving for b. These lines are seen, projecting back to the y-axis, as dotted green - partly overplotted in red. The intercepts are 4.75 and -2.3333 as shown. Now we have the equations for the two lines in the y = mx+b form.

We seek x and y at the intersection of these two lines as the center of our circle. That is we want to solve for x and y, and we do have two simultaneous linear equations – the equations of the two bisecting lines. In the program, we use matrix inversion, but solving by hand by substitution of quite easy as well. We find the center of the circle at 1.9318 and 0.8864. The radius, 2.1573, is just the distance from the center to any of the original points (we use Pythagoras in the program). That's it.

If we apply this to survey maps, it will frequently be used for boundary sections that just ended up a slight bit curved. In such a case, we might have endpoints, and we find a marker between that is slightly off. Thus the two chords will be almost on a straight line, and accordingly, the radius to the center point is a long way off – like over in another block. That is, it wouldn't look much like Fig. 4. This could mean that the surveyor's compass is not wide enough to plot the curve – hence the advantage of a computerized drawing program.

# PROGRAM *fitcircle.m*

```
function [cx,cy,r] = fitcircle(x1, y1, x2, y2, x3, y3)
% xc, yc = center of circle   r = radius
% input x and y of three points to fit
% if divide by zero error - move a point evers o slightly - easy fix

% find two straight lines a and b:  y = slope*x + intercept
slopea=(y2-y1)/(x2-x1)
slopeb=(y3-y2)/(x3-x2)
intercepta=(y1*x2-x1*y2)/(x2-x1)
interceptb=(y2*x3-x2*y3)/(x3-x2)

% midpoints of lines
mpax = (x2+x1)/2
mpay = slopea*mpax + intercepta
mpbx = (x3+x2)/2
mpby = slopeb*mpbx + interceptb

% bisector slopes
bislopea=-1/slopea
bislopeb=-1/slopeb
% bisector intercepts
biintercepta= mpay - bislopea*mpax
biinterceptb= mpby - bislopeb*mpbx
% matrix of two equations
A=[-bislopea 1;-bislopeb 1]
% invert to solve
center=inv(A)*[biintercepta biinterceptb]'
% center and radius
cx=center(1)
cy=center(2)
r=sqrt( (x1-cx)^2 + (y1-cy)^2)
```

```
% Now Plot it All

figure(1)
plot([x1 x2],[y1 y2])
hold on
plot(mpax,mpay,'o')
plot([x2 x3],[y2 y3])
plot(mpbx,mpby,'o')
plot([x1 x2 x3],[y1 y2 y3],'r*')
plot([0 mpax],[biintercepta mpay],'g:')
plot([0 mpbx],[biinterceptb mpby],'g:')
plot(cx,cy,'ok')
plot(cx,cy,'*k')
ang=0:3599;
ang=2*pi*ang/2600;
circlex=r*sin(ang)+cx;
circley=r*cos(ang)+cy;
plot(circlex,circley,'k')
plot([mpax cx],[mpay cy],'r')
plot([mpbx cx],[mpby cy],'r')
axis equal
axis([-2 5 -3 5])
grid
hold off
figure(1)
```