Surveying - Traverse Closure
by Louie L. Yaw
Walla Walla University
June 27, 2019

key words: traverse, closure, departures, dot product, angles, coordinates, surveying

1 Introduction

In surveying it is often necessary to do a traverse closure. Basically, a polygon is surveyed in the field. The polygon contains n sides. Each polygon side has length, L_i. Between each side is an interior angle α_i. In general, the surveyed lengths and interior angles contain error. The sum of the interior angles should be $(n-2)180$, however, in general the surveyed angles do not exactly sum to this value. Hence, the angles must be adjusted. Based on these corrected angles the lengths of the polygon sides have x components and y components. Since the polygon is closed the sum of x components must equal zero and the sum of y components must equal zero. In general these sums are not exactly equal to zero. Hence, the x and y components are adjusted to account for the error. When both sum of angles and sums of components equal the required values the traverse closure is done. The result is an adjusted polygon that has angles and lengths that are geometrically consistent with the closed polygon. This article describes a process by which this may be accomplished using a MATLAB script.

2 An example problem

Consider the polygon of Figure 1 with surveyed lengths and interior angles shown. Notice the polygon contains number of sides $n = 7$, lengths L_i, interior angles α_i, exterior angles β_i, nodes $i = 1$ to 7, and node coordinates x_i, y_i. It is important to recognize that the ensuing algorithms are based on the sides, angles, nodes, being ordered as shown in counterclockwise order. Furthermore, to start, side 1 is chosen to coincide with the x-axis of a Cartesian coordinate system with node n at the origin. In the following section the steps to accomplish a traverse closure are provided for the example problem. The only starting inputs necessary are the side lengths, L_i, and interior angles, α_i.

3 Traverse Closure

Each step in the traverse closure process is provided in its own subsection below.

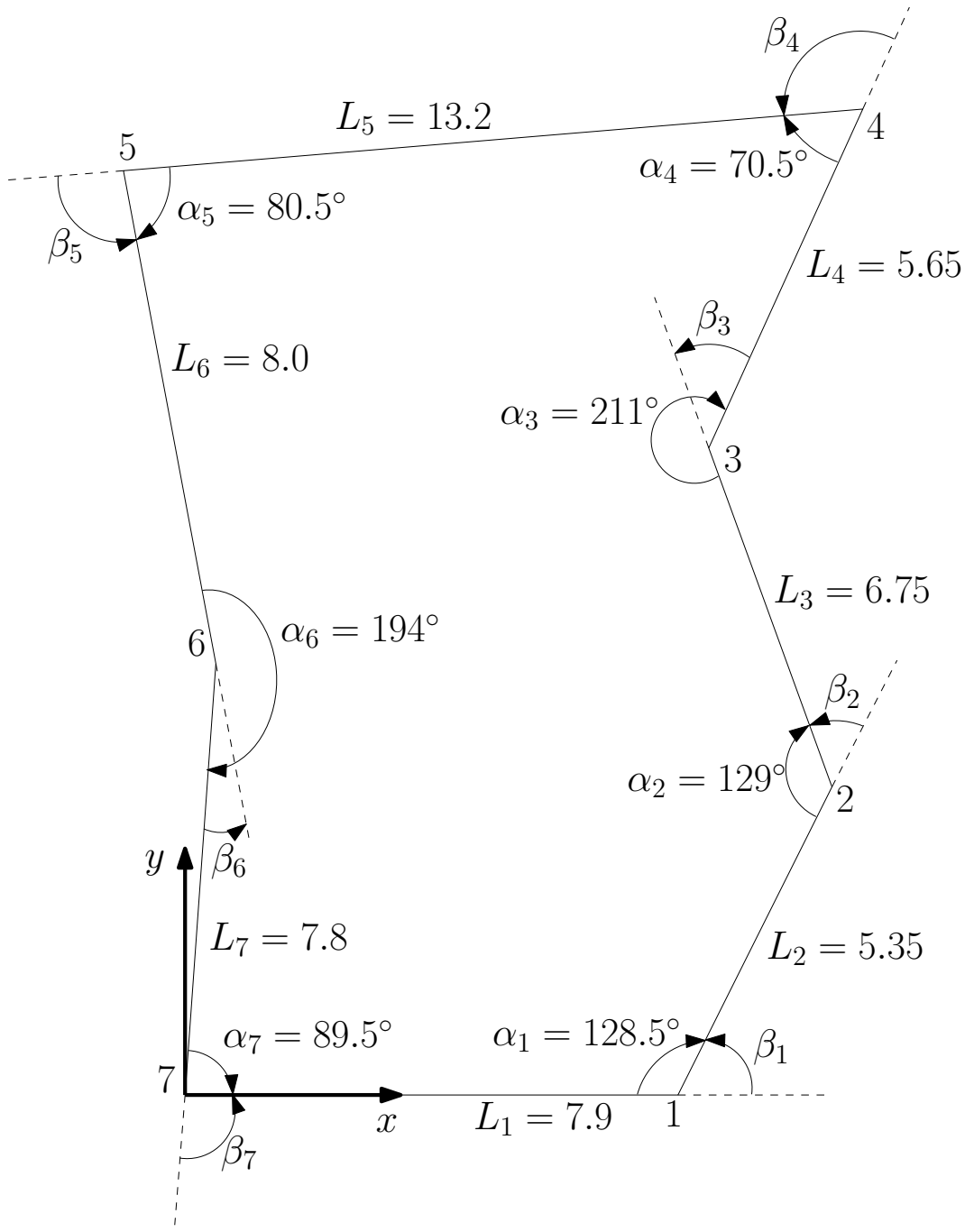

Figure 1: Example problem surveyed polygon (not to scale)

3.1 Adjusting the angles, α_i

For an n sided polygon the sum of the interior angles in degrees should equal $(n-2)*180°$. For the example problem,

$$\sum_{i=1}^{n} \alpha_i = (128.5 + 129 + 211 + 70.5 + 80.5 + 194 + 89.5) = 903°. \tag{1}$$

The sum of angles for the example 7 sided polygon should be

$$(n-2)*180 = (7-2)*180 = 900°. \tag{2}$$

Clearly, the error is

$$error = (n-2)180 - \sum_{i=1}^{n} \alpha_i = 3°. \tag{3}$$

To adjust the angles add $error/n$ to each angle. This is shown in Table 1.

i	Angle, α_i	Correction	Adjusted α_i
1	128.5	-0.42857	128.071
2	129.0	-0.42857	128.571
3	211.0	-0.42857	210.571
4	70.5	-0.42857	70.071
5	80.5	-0.42857	80.071
6	194.0	-0.42857	193.571
7	89.5	-0.42857	89.071

Table 1: Adjusting angles, α_i

3.2 Calculate angles, β_i

Notice each angle β_i is found by taking $180° - \alpha_i$, where the updated adjusted α_i are used. The result is a positive angle if β_i is outside the polygon or is a negative angle if β_i is inside the polygon as seen in Figure 1. That is

$$\beta_i = 180 - \alpha_i. \tag{4}$$

The resulting values of β_i are shown in Table 2.

3.3 Calculate angles, γ_i

Polygon side i is described as pointing from node $i-1$ to node i. The angular direction the side points, counterclockwise from the positive x-axis, is a calculated angle, γ_i.

$$\gamma_i = \sum_{k=1}^{i} \beta_k \tag{5}$$

i	Angle, β_i
1	51.9286
2	51.4286
3	-30.5714
4	109.9286
5	99.9286
6	-13.5714
7	90.9286

Table 2: Angles, β_i

Notice γ_i is simply the sum of all β_k values up to and including node i. The angle γ_i provides the polar coordinates direction of line segment i of the polygon. Notice that these angles always have value between 0 and 360° regardless of the number of polygon sides. In fact, the correct value of γ_n should always be 360° (after prior α_i adjustments are made). Table 3 contains the resulting γ_i values for the example problem.

i	Angle, γ_i
1	51.9286
2	103.3571
3	72.7857
4	182.7143
5	282.6429
6	269.0714
7	360.0000

Table 3: Angles, γ_i

3.4 "Departures" (components) for each polygon side i

Having this angle it is easy to calculate the "departures" (components) dx_i and dy_y in rectangular cartesian coordinates for polygon side i. These "departures" have positive or negative signs depending on the direction polygon side i is pointing. Since polygon side 1 is chosen to point in the x-direction it has departures $dx_1 = L_1$ and $dy_1 = 0$. For all remaining sides the departures (Table 4) are calculated as

$$\begin{aligned} dx_i &= L_i \cos \gamma_{i-1} \\ dy_i &= L_i \sin \gamma_{i-1}. \end{aligned} \tag{6}$$

3.5 Departure errors

If a polygon is closed then the sum of all x departures (positive and negative ones) should equal zero, and similarly for y departures. If they do not sum to zero then the final sum is

i	dx_i	dy_i
1	7.9000	0.0000
2	3.2990	4.2117
3	-1.5594	6.5674
4	1.6721	5.3969
5	-13.1852	-0.6251
6	1.7510	-7.8060
7	-0.1264	-7.7990

Table 4: Departures

the error. That is

$$\begin{aligned} \sum_{i=1}^{n} dx_i &= error_{dx} \\ \sum_{i=1}^{n} dy_i &= error_{dy}. \end{aligned} \tag{7}$$

3.6 Adjustment of departures

The d_x and d_y departures are adjusted to account for the departure errors. It is assumed that the errors are disbursed amongst the departures in proportion to the length of the departure. The adjusted departures (Table 5) are calculated according to the following formulas:

$$\begin{aligned} dx_i &= dx_i - \left(\frac{|dx_i|}{\sum_{i=1}^{n} |dx_i|} \right) error_{dx} \\ dy_i &= dy_i - \left(\frac{|dy_i|}{\sum_{i=1}^{n} |dy_i|} \right) error_{dy}. \end{aligned} \tag{8}$$

i	dx_i	dy_i
1	7.9667	0.0000
2	3.3269	4.2188
3	-1.5462	6.5784
4	1.6862	5.4059
5	-13.0739	-0.6241
6	1.7658	-7.7930
7	-0.1253	-7.7860

Table 5: Adjusted Departures

3.7 Calculation of nodal coordinates

The coordinates for each node (Table 6) are calculated from the adjusted departures according to the following formulas.

$$\begin{aligned} x_i &= \sum_{k=1}^{i} dx_k \\ y_i &= \sum_{k=1}^{i} dy_k. \end{aligned} \tag{9}$$

i	x_i	y_i
1	7.9667	0.0000
2	11.2935	4.2188
3	9.7473	10.7971
4	11.4335	16.2030
5	-1.6404	15.5790
6	0.1253	7.7860
7	0.0000	0.0000

Table 6: Nodal Coordinates

3.8 Inferred angles, δ_i

From the adjusted departures the nodal coordinates are found. The nodal coordinates in turn infer (or imply) the angles, δ_i. It is now necessary to calculated the inferred angles and see if their sum agrees with the correct sum of interior angles for a polygon with n sides. Hence, δ_i values are calculated from the nodal coordinates. This is accomplished as follows.

For a particular node i two line segments emanate from the node. From the coordinates construct vectors along each of the line segments. Each vector is constructed with its tail located at node i. For example, for node 2.

$$\begin{aligned} \boldsymbol{v} &= (x_1 - x_2)\boldsymbol{i} + (y_1 - y_2)\boldsymbol{j} = -3.3269\boldsymbol{i} - 4.2188\boldsymbol{j} \\ \boldsymbol{w} &= (x_3 - x_2)\boldsymbol{i} + (y_3 - y_2)\boldsymbol{j} = -1.5462\boldsymbol{i} + 6.5784\boldsymbol{j}. \end{aligned} \tag{10}$$

For this case, for node 2, the inferred angle(s) between the vectors are

$$\begin{aligned} \delta_{2a} &= \arccos \frac{\boldsymbol{v} \bullet \boldsymbol{w}}{vw} = 128.5139^\circ \\ \delta_{2b} &= 360 - \delta_{2a} = 360 - 128.5139 = 231.4861^\circ. \end{aligned} \tag{11}$$

Of the two answers the one that is closest to the most recently calculated $\alpha_2 = 128.5139$ is the correct value. Hence, in this case the final inferred angle is $\delta_2 = 128.5139$. This process is easily accomplished algorithmically within a MATLAB script for all inferred angles. Finally, the inferred δ_i angles *are* the new α_i values. Therefore, the α_i are set equal to the δ_i. As a result, in Table 7 the final inferred α_i values are shown.

i	Angle, $\alpha_i = \delta_i$
1	128.2589
2	128.5139
3	210.5508
4	69.9435
5	79.9661
6	193.6889
7	89.0777

Table 7: Inferred Angles, $\alpha_i = \delta_i$

3.9 Calculated side lengths

The side lengths (Table 8) of the polygon are easily calculated by the most recent (adjusted) departures. This is accomplished according to the Pythagorean formula

$$L_i = \sqrt{dx_i^2 + dy_i^2}. \tag{12}$$

i	L_i
1	7.9667
2	5.3727
3	6.7576
4	5.6628
5	13.0888
6	7.9906
7	7.7870

Table 8: Polygon side lengths, L_i

3.10 Comments

The angles of Table 7 sum to 900° as they should. If the angles are used as before to eventually calculate the x and y departures it is found that the departure errors are very small to machine precision. It is evident that the procedure above gives direct results and no iteration is required.

3.11 MATLAB script

As indicated the above procedure is amenable to solution in a computer. A MATLAB script (traverse.m) and a function (dotproductangle.m) are included with this paper in the appendix. These scripts quickly and easily complete the traverse closure procedure when given the appropriate input.

4 Conclusion

A procedure for traverse closure is presented. MATLAB scripts are provided to automate the process in a computer. No iterations are required and satisfactory results to machine precision are obtained.

5 Appendix

5.1 MATLAB script traverse.m

```
%traverse1.m by Louie L. Yaw 6/25/19
%Script to find the corrected angles and lengths to do a traverse closure
%for surveying.

%Bryc Cole Test case for verification

%Begin user input
n=3; %Number of traverse lengths
%Initialize variables
alpha=zeros(n,1);%interior polygon angles
length=zeros(n,1);

%Input of surveyed values (in counterclockwise order)
%alpha=[67;57.317;55.683];
%length=[147.65;147.23;163.19];
alpha=[67.0882;56.8906;55.8375];
length=[148.26;146.78;163.0327];

%Begin calculations

%Check angles residual
aResid=(n-2)*180-sum(alpha);

%Correct the angles
acorrect=(aResid/n)*ones(n,1);
alpha=alpha+acorrect;

%Assume Line1 points along x direction, (x=East West, y=North South)
dx=zeros(n,1);
dy=zeros(n,1);
%Find beta angles
beta=zeros(n,1);%in degrees
for i=1:n
    beta(i)=180-alpha(i);
end
```

```
%Find gamma angles for each line i
gamma=zeros(n,1);%in degrees
for k=1:n
for i=1:k
    gamma(k)=gamma(k)+beta(i);
end
end

%Find the x and y departures
dx(1)=length(1);%since length 1 is entirely in the x direction and dy(1)=0;
for i=2:n
  dx(i)=cos(pi/180*gamma(i-1))*length(i);
  dy(i)=sin(pi/180*gamma(i-1))*length(i);
end

%Find the x and y departure errors
dxerror=sum(dx);
dyerror=sum(dy);

%Adjust the x and y departures
dxabssum=sum(abs(dx));
dyabssum=sum(abs(dy));
for i=1:n
  dx(i)=dx(i)-abs(dx(i))/dxabssum*dxerror;
  dy(i)=dy(i)-abs(dy(i))/dyabssum*dyerror;
end

%Find coordinates of all points
x=zeros(n,1);
y=zeros(n,1);
for i=1:n
  if i==1
    x(i)=dx(i);
    y(i)=dy(i);
  else
    x(i)=dx(i)+x(i-1);
    y(i)=dy(i)+y(i-1);
  end
end

%Find the inferred angles, delta, based on coordinates and dot product
delta=zeros(n,1);
for i=1:n
  if i==1
```

```
    delta(i)=dotproductangle(x(n),y(n),x(i),y(i),x(i+1),y(i+1));
  elseif i==n
    delta(i)=dotproductangle(x(n-1),y(n-1),x(i),y(i),x(1),y(1));
  else
    delta(i)=dotproductangle(x(i-1),y(i-1),x(i),y(i),x(i+1),y(i+1));
  end
end
%Due to ambiguity of cos angles, correct the inferred angle, delta, to be
%one closest with corrected angle, alpha
for i=1:n
  a1=delta(i);
  a2=360-delta(i);
  r1=abs(a1-alpha(i));
  r2=abs(a2-alpha(i));
  if r1<r2
    delta(i)=a1;
  else
    delta(i)=a2;
  end
end
alpha=delta;
%Find new lengths
for i=1:n
  if i==1
    length(i)=sqrt((x(n)-x(i))^2+(y(n)-y(i))^2);
  else
    length(i)=sqrt((x(i-1)-x(i))^2+(y(i-1)-y(i))^2);
  end
end

display('Output')
alpha
length
```

5.2 MATLAB script dotproductangle.m

```
%dotproductangle.m
%by Louie L. Yaw 6-26-19
%Function find dotproductangle given input
%
%Construct dotproductangle function === dotproductangle(xnm1,ynm1,xn,yn,xnp1,ynp1)
%
%where,
%
%xnm1=coordinate x_n-1
```

```
%ynm1=coordinate y_n-1
%xn=coordinate xn at which the interior angle is to be determined
%yn=coordinate yn at which the interior angle is to be determined
%xnp1=coordinate x_n+1
%ynp1=coordinate y_n+1

function angle=dotproductangle(xnm1,ynm1,xn,yn,xnp1,ynp1)

%Create dotproductangle function
L1=sqrt((xn-xnm1)^2+(yn-ynm1)^2);
L2=sqrt((xn-xnp1)^2+(yn-ynp1)^2);
v1=[xnm1-xn;ynm1-yn];
v2=[xnp1-xn;ynp1-yn];
angle=acos(v1'*v2/(L1*L2))*180/pi;%in degrees

%End of the function dotproductangle.m
```