PROGRAMMING THE FREE ADJUSTEMENT OF THE PRECISE LEVELING MEASUREMENTS

Călin Alexandru, Ph.D., Lecturer, Technical University of Civil Engineering, Faculty of Geodesy, alexandru.calin@geodezie.utcb.ro
Gînța Alexandru, eng.,masters degree student at Technical University of Civil Engineering, Faculty of Geodesy, ginta_alexandru@yahoo.com
Dumitru Paul Daniel, Ph.D., Lecturer, Technical University of Civil Engineering, Faculty of Geodesy, paul.dumitru@geodezie.utcb.ro

Abstract: *This work presents a study case about the free adjustment of the precise leveling measurements using an application designed with the help of the program DELPHI. The article presents the mathematical model of the method of the least squares, Gauss – Markov, and the phases of achieving our proposed problem. For a data set consisting in: measurements of leveling, lengths of the leveling lines and provisional values for the unknowns (elevation), we managed, by processing, to determine the most probable values for the elevations with the accuracy of their determination and the most probable values for the measurements, of course with their associated accuracy. Also, the article presents the application developed in DELPHI [6], using PASCAL programming language.*

Keywords*: free adjustment, leveling network, precise leveling, Delphi, programming*

1. Introduction

The current work presents the steps taken to create a program that automatically processes the measurements from a leveling network.

An altimetric network is a network in which points are defined by a single coordinate. (1D Space).

Following the mathematical model of the least squares method (Gauss - Markov) we have succeeded to obtain, through processing, the most probable values for the elevations, along with their accuracy and the most probable values for our measurements, of course with their associated accuracy. All of these where obtained starting from a set of datas consisting in: measurements of the diferences in level, leveling line lenghts and provisional values.

2. Mathematical support

The functional – stochastic model for the indirect measurement method [1] is (2.1 si 2.2):

$$V = Ax + l \tag{2.1}$$

$$C_m = \sigma_0^2 * Q_m \tag{2.2}$$

In preparing the functional model, we used the correction equation for geometric leveling networks, the general case [1] (2.3):

$$v_{ij} = x_j - x_i + H_j^0 - H_i^0 - \Delta h_{ij}^* ,$$
$$v_{ij} = x_j - x_i + l_{ij} ,$$
$$l_{ij} = H_j^0 - H_i^0 - \Delta h_{ij}^* . \tag{2.3}$$

where x_i and x_j are the unknowns, H_j^0, H_i^0 are the provisional elevations for the point ,j' and ,i' and v_{ij} is the coresponding correction for the measurement of the height diferences measured in leveling, Δh_{ij}^*.[1]

The matrix ,A', or the design matrix, was prepared using the unknowns coeficients from the corection equation. The elements of the design matrix will be -1, 0 and 1. However, knowing that the network is considered as a free network, to the design matrix it has been added one more line at the end, in which every element is ,1'. This is a fictive equation and represents the condition for the unkwons, the sum of the unknowns being equal to ,0'.

Weighting matrix is a diagonal matrix in which the elements were calculated by (2.4):

$$p = \frac{1}{L_{[km]}} \tag{2.4}$$

Likewise, to this matrix it has been added one more line at the end (fictive equation), in which al elements except the last one are 0, and the last element is calculated with the formula [2] (2.5):

$$p_f = 100 * \frac{1}{m} * \sum_{i=1}^{m} p_i \tag{2.5}$$

That being established, we've moved on to solving the system. The first step was the normalization of the system which involves the calculation of the normal matrix ,N' and its inverse, ,N^{-1}'(2.6).

$$N = A^t P A \tag{2.6}$$

The next stept is the calculation of the most likely values for our unknowns. In this case, these are the corections for our elevations [2].(2.7)

$$x = -N^{-1} A^t P l \tag{2.7}$$

After that the corrections for the original measurements were calculated, ie array ,v':

$$V = Ax + l \tag{2.8}$$

The last stept was to calculate the precision elements[1], [2], [4]:

a) Standard deviation of the weight unit (2.9):

$$s_0 = \pm \sqrt{\frac{[pvv]}{m - n + d}} \tag{2.9}$$

s_o – standard deviation of the weight unit
m – number of measurements
n – number of unknown elements
d – rank defect

b) Standard deviation for an unknown element (2.10):

$$s_{xi} = s_0 \sqrt{q_{ii}}, i = 1, n \tag{2.10}$$

q_{ii} – element ,i' from the main diagonal of the inverse of the normal matrix

c) Standard deviation for a processed measurement (2.11):

$$s_{mi} = \pm \frac{s_0}{\sqrt{p_i}} \tag{2.11}$$

p_i – element ,i' from the main diagonal of the weight matrix.

d) Average standard deviation for the network (2.12):

$$s_t = \pm \frac{1}{n} \sum_{i=1}^{n} s_{xi} \tag{2.12}$$

3. Logical support

Both the reading and the displaying of our data set were made from and in ,*.txt' files, named ,in.txt' and ,out.txt'.

In the ,in.txt' file we will enter the initial data in the following order:

- On line 1, the number of points which are included in the compensation process.
- On line 2, the name of the points which are included in the compensation process.
- On line 3, we will have the points name for which we have a provisional elevation and its elevation
- Starting from the 4-th line, the measurements of elevation differences will be introduced in a format like 'From' 'to' 'level difference' 'length of the leveling line', as seen in Fig. 3.1:

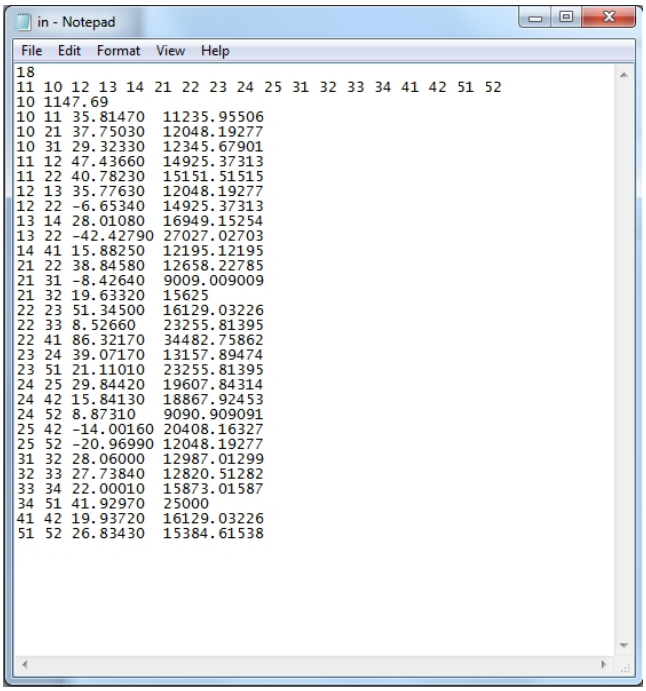


Fig. 3.1 – Model of an input file

Thus, the program will retain in a variabile ,n' the number of the unknowns, in a vector ,t' the unknowns, in a matrix named ,cota' the level of the provisional point on its position and the measurements will be stored in two arrays: ,b' and ,c'. Each of these arrays will have the number of lines equal to the number of measurements and 2 columns. For example, the array ,b' will contain the elements ,from' and ,to' on the position b[i,1] and b[i,2]. The array ,c' will contain the element c[i,1] and c[i,2]. In the first element we store the diference in level between points b[i,1] and b[i,2] and in the second element we store the lenght of the leveling line between the same points.

We chose this division because the elements from the matrix ,b' will be integer and the elements from the matrix ,c' will be real.

In Fig. 3.2 we can see some code lines that exemplifies those previously stated.

```
  //nr necunoscute
  readln(f,n);
  //necunoscutele
  for i := 1 to n do
  begin
   read(f,t[i]);
  end;
  readln(f);
//cota provizorie;
  read(f,pct);
  for i := 1 to n do if t[i]=pct then begin readln(f,cota[i,1]); q:=i; end;


//masuratorile
writeln(g,'Prelucrarea unei retele de nivelment geometric ca retea libera');
writeln(g,'-------------------------------------------');
k:=0;
repeat
 k:=k+1;
 readln(f,b[k,1],b[k,2],c[k,1],c[k,2]);
 writeln(g,'De la ',b[k,1],' la ',b[k,2],' dH*=',c[k,1]:8:4,' m L=',c[k,2]:10:3,'
until eof(f);
```

Fig. 3.2 – Input and output of the initial data

Also the initial data were displayed in the output file ‚out.txt'.

Next, the application computes the provisional elevation for all of the points.

It was stated earlier that the value for the provisional elevation was saved in an array named ‚cota'. This array has the same number of lines as the number of the unknowns and two columns. On the first column was retained the value of the elevation for the point and the second column is a counter that will take the values ‚0' or ‚1'. A value of ‚1' is given when the point has an assigned value for its provisional elevation. At first, all elements, except for the provisional one will have the element ‚0' on both columns.

The process for distributing the known provisional elevation is:

- Identification of the equation that contains a known elevation and an unknown elevation.
- Calculation of the unknowns elevation
- Assigning the value '1' for the element cota[i,2], where 'i' is the point position for which the elevation was calculated in the preceding step.

These three steps are repeated until all points receive a provisional elevation, in this case, until all elements stored in the second column of the array ‚cota' are equal to ‚1'.

This process, writen in code lines, was exemplified in Fig. 3.3

```
//distribuire cota
for i := 1 to n do cota[i,2]:=0;
cota[q,2]:=1;
repeat
contor:=0;
 for j := 1 to k do
   begin
   for i := 1 to n do begin if t[i]=b[j,1] then q:=i; end;
   for i := 1 to n do begin if t[i]=b[j,2] then w:=i; end;
     if (cota[q,2]=0) or (cota[w,2]=0) then
      begin
       if (cota[q,2]=1) and (cota[w,2]=0) then begin cota[w,1]:=cota[q,1]+c[j,1];
       if (cota[q,2]=0) and (cota[w,2]=1) then begin cota[q,1]:=cota[w,1]-c[j,1];
      end;
   end;
for q := 1 to n do if (cota[q,2]=1) then contor:=contor+1;
until contor=n;
    writeln(g);
  writeln(g,'Cotele Provizorii ale punctelor: ');
  writeln(g,'-------------------------------------------');
  for i := 1 to n do writeln(g,'Ho',t[i],'= ',cota[i,1]:9:4,' m');
```

Fig. 3.3 – Distributing the known provisional elevation

The next step was populating the design matrix ‚a' with elements 0, -1 and 1. The population was done with the help of the array ‚b'. Thus, for the line ‚i' from the array ‚b' we have the elements b[i,1] and b[i,2]; ‚from' and ‚to'. Looking at the correction equation we see that the element ‚from' has the coefficient ‚-1' and the element ‚to' has the coefficient ‚1'. The rest of the line is composed from the element ‚0'. So the application searches within the array ‚t', and if the equality b[i,1]=t[i] the value ‚-1' will be assigned to the element of the design matrix. Analog if b[i,2]=t[i] the value ‚1' will be assigned. This sequence is executed line by line, for each measurement.

Following the mathematical model we observe that a final line must be added to the design matrix. This line has all elements equal to ‚1'. The population of the design matrix was is presented in Fig. 3.4

```
//MAtrice design
q:=0;
repeat
  q:=q+1;
  r:=0;
   repeat
   r:=r+1;
    if b[q,1]=t[r] then a[q,r]:=-1;
    if b[q,2]=t[r] then a[q,r]:=1;
   until r=n ;
until (q=k);
k:=k+1;
for i := 1 to n do
    a[k,i]:=1;
```

Fig. 3.4 – Population of the design matrix

The elements of the weight array and the free elements array were calculated according to the mathematical support, as seen in Fig. 3.5.

```
//matricea ponderilor
//writeln(g,'Matricea ponderilor:');
for i := 1 to k-1 do
begin
 p[i,i]:=1/(c[i,2]/1000);
 p[k,k]:=p[k,k]+p[i,i];
 end;
 p[k,k]:=p[k,k]*100/(k-1);
writeln(g);
//matricea termenilor liberi
//writeln(g,'Matricea termenilor liberi');
  for j := 1 to k-1 do
  begin
    for i := 1 to n do begin if t[i]=b[j,1] then q:=i; end;
    for i := 1 to n do begin if t[i]=b[j,2] then w:=i; end;
    l[j,1]:=cota[w,1]-cota[q,1]-c[j,1];
    //writeln(g,l[j,1]:8:5);
  end;
```

Fig. 3.5 – Calculus of the weight array and the free elements array

With those arrays estabilished we procedeed to the calculus of the arrays: A^tP, A^tPA(the normalized array, named in this case ‚m') and the array A^tPl.

The next step is to inverse the matrix ‚m'. Because there is no code sequences that automaticaly inverses an array, we had to use the **Lower – Upper Decomposition[3].**

Lower – Upper Decomposition is a calculus method based on the assumption that a square matrix can be decomposed into a product of two matrices: a lower triangular matrix and a upper triangular matrix. So, any given matrix, ‚a' can be written as:

$A=T*S$ (3.1)

T – lower triangular matrix,

S – upper triangular matrix.

The lower/upper triangular matrix is a square matrix in which all the elements above or below the main diagonal are 0.

This method to calculate the inverse of a matrix was used because:

$A^{-1}=(T*S)^{-1}=S^{-1}*T^{-1}$ (3.2)

It is much easier to inverse a lower or upper triangular matrix rather then a full sqare matrix.

For the factorization we followed CROUT's algorithm [5]:

- To the elements from the main diagonal of the lower triangular matrix value 1 is assigned.

  t[i,i]:=1; i=1,n; (3.3)
- The elements from the first line of the upper triangular matrix are equal to the elements from the first line of the initial matrix.

  s[1,j]:=a[1,j], j=1,n; (3.4)
- Elements t[i,1] are calculated with the formula 3.5:

  t[i,1]=a[i,1]/s[1,1]; i=2,n; (3.5)
- For each p, p=1…n, first we compute the elements from line 'p' of the upper triangular matrix with formula:

$$s[p,j]=a[p,j]-\sum_{k=1}^{p-1}(t[p,k]s[k,j]);\quad j=p,n \qquad (3.6)$$

and then the elements of the column ‚p' from the lower triangular matrix with formula:

$$t[i,p]=1/s[p,p]\ (a[i,p]-\sum_{k=1}^{p-1}(t[i,k]s[k,p]))\quad i=p+1,n \qquad (3.7)$$

In this case the lower triangular matrix is named ‚inf' and the supperior triangular matrix is named ‚sup'. Thus resulting, after LU Factorization that: 'm=inf*sup'. The LU decomposition writen in code lines was prezented in Fig. 3.6.

```
// Factorizare lower upper
for i := 1 to n do
 inf[i,i]:=1;
for j := 1 to n do
  sup[1,j]:=m[1,j];
for i := 2 to n do
  begin
    inf[i,1]:=m[i,1]/sup[1,1];
  end;
for p1 := 2 to n do
  begin
    for j := p1 to n do
      begin
        aux:=m[p1,j];
        for k1 := 1 to p1-1 do
          begin
            aux:=aux-inf[p1,k1]*sup[k1,j];
          end;
        sup[p1,j]:=aux;
      end;
   if p1<>n then
   begin
    for i := p1+1 to n do
      begin
        aux:=m[i,p1];
        for k1 := 1 to p1 - 1 do
          begin
            aux:=aux-inf[i,k1]*sup[k1,p1];
          end;
        inf[i,p1]:=aux/sup[p1,p1];
      end;
   end;
  end;
```

Fig. 3.6 – LU Decomposition

Next we calculated the inverse of the lower triangular matrix and the upper triangular matrix using the fact that if A is an invertible matrix and B is its inverse then:

A*B=I (3.8)

But in our case the array, to be inverted, is triangular, thus its inverse will also be a triangular array. So, we have deduced a formula to calculate every element from the inverse array.

$$\begin{pmatrix} a11 & a12 & a13 & \dots & a1n \\ 0 & a22 & a23 & \dots & a2n \\ 0 & 0 & a33 & \dots & a3n \\ & & \dots & & \\ 0 & 0 & 0 & 0 & ann \end{pmatrix} * \begin{pmatrix} b11 & b12 & b13 & \dots & b1n \\ 0 & b22 & b23 & \dots & b2n \\ 0 & 0 & b33 & \dots & b3n \\ & & \dots & & \\ 0 & 0 & 0 & 0 & bnn \end{pmatrix} = I$$

(3.9)

It is obvious that the elements from the main diagonal of the inverse can be computed with the formula:

$$b[i,i] = \frac{1}{a[i,i]}, \text{ i=1,n;} \quad (3.10)$$

Having the elements of the main diagonal calculated, we can calculate the rest of the elements starting from those which are located directly above the main diagonal, after that those which are located above the last elements calculated and so on. The formula is:

$$b[i,j] = -\frac{1}{a[i,i]} * \sum_{q=i+1}^{j} (a[i,q] * b[q,j]) \quad i,j := 1 \dots n; \tag{3.11}$$

This formula will work if j>1 (in this case, the matrix is a upper triangular one) and the diference between the column counter and the line counter is 1. (we are situated above the elements previously calculated).

The same goes for a lower triangular matrix.

Moving on, we obtained the matrices ‚inv_inf', the inverse of the lower triangular matrix, and ‚inv_sup', the inverse of the upper triangular matrix thus resulting, by multiplicating the last two, the inverse of the normal matrix, ‚m'

Next, the program calculates the correction arrays, ‚x' and ‚v' by multipling the arrays that composes them, acording to the mathematical support.

The last stept is to calculate the precision elements using the formulas from the mathematical support.

All the output datas are writen in the file ‚out.txt'.

4. Conclusions

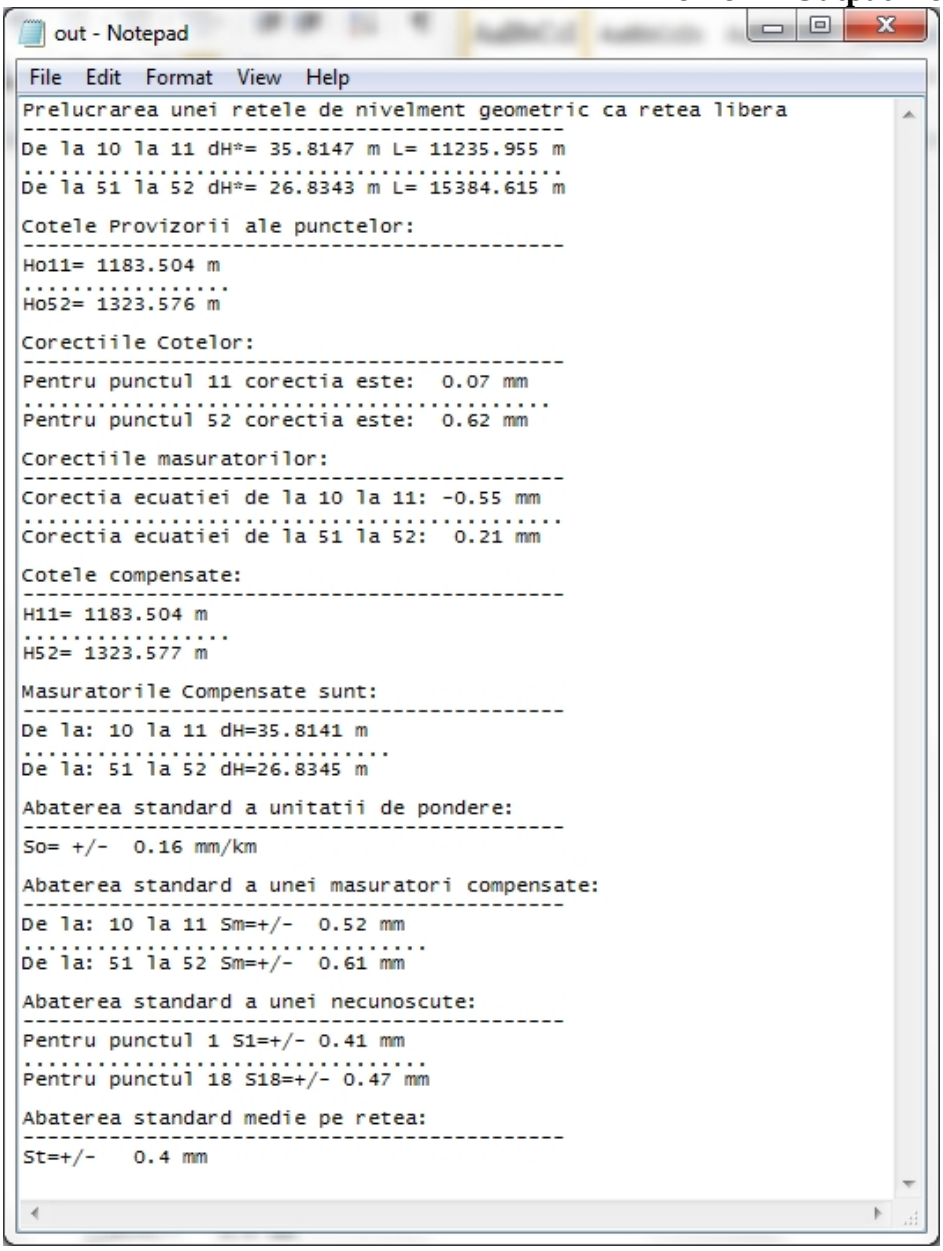
The aim of the study was to obtain an automated method for processing the measurements from a geometric leveling network. This was done using both knowledge of geodesy (leveling networks, processing of the measurements from a leveling network) and programming (programming language Pascal, Delphi).

After processing, the program offers the solutions: the most probable values for the elevations, standard deviation of determination for elevations, the most probable values for the leveling measurements and the standard deviations for a measurement. (Anexxe 1 – Output file). Also, the application ca be used in further research and development, for processing networks for deformations studying and it can run under another program**.**

References

1. *Moldoveanu, C: "Geodezie Matematică", Lecture Notes, Technical University of Civil Engineering, Faculty of Geodesy, 2010 - 2011*
2. *Moldoveanu, C: "Geodezie", Ed. Matrixrom,2002, Bucureşti,*
3. *Moldoveanu, C: "Sisteme Informatice în Măsurători Terestre", Lecture Notes, Technical University of Civil Engineering, Faculty of Geodesy, 2010 – 2011*
4. *Danciu, V: "Compensarea Măsurătorilor şi Statistică", Lecture Notes, Technical University of Civil Engineering, Faculty of Geodesy, 2010 – 2011*
5. *Călin, A: "Prelucrarea automata a datelor geodezice", Lecture Notes, Technical University of Civil Engineering, Faculty of Geodesy, 2010 – 2011*
6. *Dehlphi user guide: http://docwiki.embarcadero.com/RADStudio/XE5/en/Delphi_Reference*

Anexxe 1 – Output file 'out.txt'



out - Notepad

File Edit Format View Help

```
Prelucrarea unei retele de nivelment geometric ca retea libera
-----------------------------------------------
De la 10 la 11 dH*= 35.8147 m L= 11235.955 m
.....................................................
De la 51 la 52 dH*= 26.8343 m L= 15384.615 m

Cotele Provizorii ale punctelor:
-----------------------------------------------
Ho11= 1183.504 m
.................
Ho52= 1323.576 m

Corectiile Cotelor:
-----------------------------------------------
Pentru punctul 11 corectia este:  0.07 mm
.............................................
Pentru punctul 52 corectia este:  0.62 mm

Corectiile masuratorilor:
-----------------------------------------------
Corectia ecuatiei de la 10 la 11: -0.55 mm
.............................................
Corectia ecuatiei de la 51 la 52:  0.21 mm

Cotele compensate:
-----------------------------------------------
H11= 1183.504 m
.................
H52= 1323.577 m

Masuratorile Compensate sunt:
-----------------------------------------------
De la: 10 la 11 dH=35.8141 m
.............................
De la: 51 la 52 dH=26.8345 m

Abaterea standard a unitatii de pondere:
-----------------------------------------------
So= +/-  0.16 mm/km

Abaterea standard a unei masuratori compensate:
-----------------------------------------------
De la: 10 la 11 Sm=+/-  0.52 mm
.................................
De la: 51 la 52 Sm=+/-  0.61 mm

Abaterea standard a unei necunoscute:
-----------------------------------------------
Pentru punctul 1 S1=+/- 0.41 mm
.................................
Pentru punctul 18 S18=+/- 0.47 mm

Abaterea standard medie pe retea:
-----------------------------------------------
St=+/-   0.4 mm
```