Algorithm for Dynamic Formation of Condition Equations in N-Dimensional Network

C. P. E. Agbachi

Department of Mathematical Sciences, Kogi State University
Anyigba, Kogi State, Nigeria

***Abstract*—** *Condition equations, once used to be the only viable and memorable option in the model of computation by least squares estimation. That was in the early days of computing when RAM size of machines was in the region of 128K and search for models with efficient use of memory was imperative. Even today the merits of condition equations remain patent. The drawback has always been the ability to form the prerequisites in the first place. This paper discusses versatile algorithms for automatic generation of condition equations, and in the process restoring the model as a utility in resolving networks in Geomatics Engineering.*

Keywords—Levels, Traverse, GPS, Cycles, Graph, BFS, DFS, Span Tree.

I. INTRODUCTION

Condition equations arise out of geometric considerations that result from evaluating internal and external consistency in a survey network.A case in point is one dimensional situation where Y is a measurable variable and of the form Y = {ΔH}.

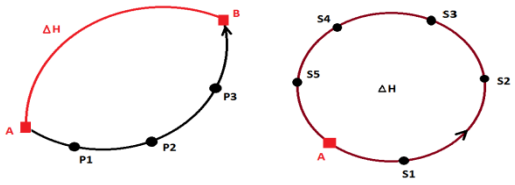

Fig. 1 Consistency in Networks

In Fig.1 is an illustration of closures in networks. The first involving a survey from A to B, fixed stations, should be in agreement with existing framework. On the second option, the internal consistency where the survey starts from and ends at the same point A, ought to be satisfied.

Enumerating in each case, there is one condition equation to fulfil:

$$hB - hA = \Delta H \text{ and } \Delta H = 0$$

Thus in simple adjustment process, the errors are spread equally between the observed stations. This translates into proportional distribution of closures over the number of stations for final values of computed positions.

A. Least Squares Approach

A rigorous solution by least squares method is available, based on condition equations. It is the option in cases of over determined equations.

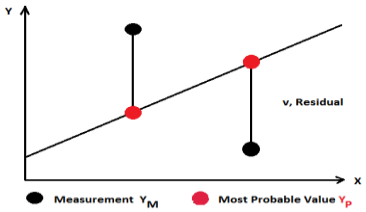

Fig. 2 Residuals

Imagine measured points Y_M as in the diagram Fig. 2, where Y_P are the most probable values such that the residual $\vartheta = Y_P - Y_M$ ----- (1)

Then if there are s observed quantities and r condition equations, r<s, the linear relationship may be expressed as follows:

$$\begin{aligned} a_{11}Y_{P1} + a_{12}Y_{P2} + \cdots + a_{1s}Y_{Ps} &= C_1 \\ a_{21}Y_{P1} + a_{22}Y_{P2} + \cdots + a_{2s}Y_{Ps} &= C_2 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots & \\ a_{r1}Y_{P1} + a_{r2}Y_{P2} + \cdots + a_{rs}Y_{Ps} &= C_r \end{aligned}$$

The above equations are idealistic in nature. In the field, the measurements form the basis of adjustment computations. Hence substituting for $Y_P = \vartheta + Y_M$,leads to $\mathbf{A\vartheta = b}$ ----- (2)

Minimising $\boldsymbol{\vartheta}^T\mathbf{W}\boldsymbol{\vartheta}$ is equivalent to minimising $\boldsymbol{\vartheta}^T\mathbf{W}\boldsymbol{\vartheta}$ $-2(\mathbf{A}\boldsymbol{\vartheta} - \mathbf{b})^T\mathbf{k}$, where **k** is the r x 1 matrix of Lagrange coefficients. This leads to $\boldsymbol{\vartheta} = \mathbf{W}^{-1}\mathbf{A}^T\mathbf{k}$.With $\mathbf{k} = (\mathbf{AW}^{-1}\mathbf{A}^T)^{-1}\mathbf{b}$ as in [1], the resulting solution is $\boldsymbol{\vartheta} = \mathbf{W}^{-1}\mathbf{A}^T(\mathbf{AW}^{-1}\mathbf{A}^T)^{-1}\mathbf{b}$ ----- (3)

1. ***Evaluation:***

Taking the models described in Fig 1, it is easy to verify that a solution exists by method of least squares condition equations. Consider the case of internal consistency where a survey starts from Station A and closes at the same point in a levelling process.

The measurements are h_1, h_2, … , h_6 and by arguments in (1),$\vartheta = h_P - h_M$. Thus as in (2)

$$(1, 1, 1, 1, 1, 1) \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix} = \Delta H$$

Let W = I, then (3) reduces to $\boldsymbol{\vartheta} = \mathbf{A^T(AA^T)^{-1}b}$.

$$\text{Hence} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix} = \begin{pmatrix} \Delta H/6 \\ \Delta H/6 \\ \Delta H/6 \\ \Delta H/6 \\ \Delta H/6 \\ \Delta H/6 \end{pmatrix}$$

And the result is $h_P = \vartheta + h_M$ which corroborates the proportional distribution of errors in cases of one condition equations.

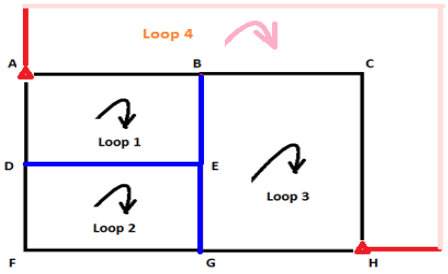

Fig. 3

In general, surveys involve networks where several conditions arise. In Fig 3for instance there are four cycles for bases of condition equations.

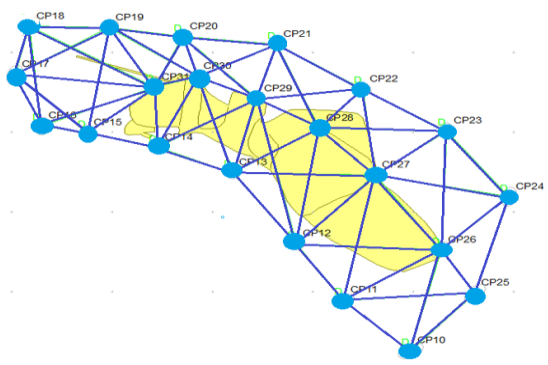

Fig. 4 Survey Network

While the number of conditions can be obvious by inspection, Fig. 3, it is not always clear in large survey networks, as in Fig. 4. Hence, the search is necessary for a suitable algorithm.

II. Graph Algorithms

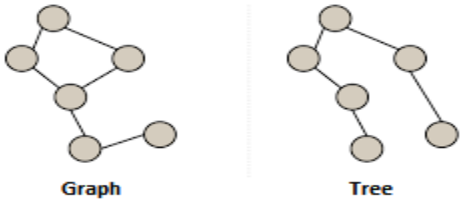

Fig. 5

A graph, Fig. 5, is a data structure important in representing information with regards to patterns, directions and in effect, networks. By definition, a graph comprises of nodes and connections known as edges. Where directions define the edges, it is known as a Digraph. Along with weighted graphs, they find common applications in Geoinformatics.

Trees and graphs are complements to each other. A tree is a connected graph without cycles. Similarly, a graph is a free tree, a tree without root. Traversal routines exist to transform graphs into spanning trees and vice versa. This is particularly useful when processing fundamental cycles [2], in order to establish condition equations.

A. Cycle Processing

By description a graph maybe defined as acyclic [3], meaning there are no cycles. However in context of survey applications, the emphasis is on adaptation, recognition of geometry and topology. Therefore, a cycle or loop starts from a node, passing through other nodes and terminating at the start node. Thus, while Fig. 3 may represent a DAG, the entities, Loop 3 etc. can be reduced to cycles by reversing observed directions in affected segments.

Processing cycles starts with a spanning tree and addition of cross or back edges. The key algorithms are Breadth First Search and Depth First Search [4].

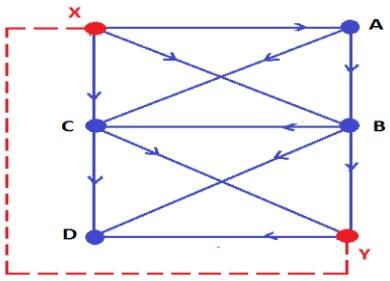

Fig. 6

In Fig 6 is a model diagram of a field survey. Typically, work originates from a control point and

terminates like wise. Thus, observed directions are important in computations and are reflected in processing algorithms. With respect to the model, the survey involves transfer of position from X, proceeding and closing at Y. Further observationsallow determination of C and D, to complete the network as a connected graph.

1. *Breadth First Search:*

Breadth First Search starts by investigating each direction emanating from X, the first station. It then moves to next layer, the next station B until finally all the nodes are connected in a spanning tree, Fig 7.

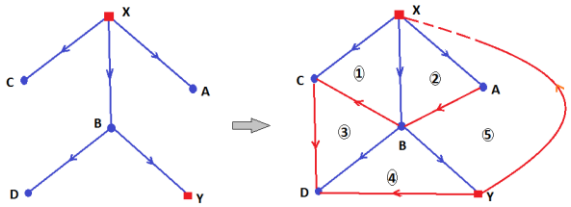

Fig. 7

Next fundamental cycles are constructed by adding cross edges. There are 5 cycles where the fifth, XBYX ensures consistency with external framework.

2. *Depth First Search:*

Depth First Search, operating from the first station X follows a one-way route in the main direction of survey, until the last stop D, when there is no further path to proceed. It then retraces its way backwards to investigate unexplored routes, BC, until completion that leads to a spanning tree, Fig 8.

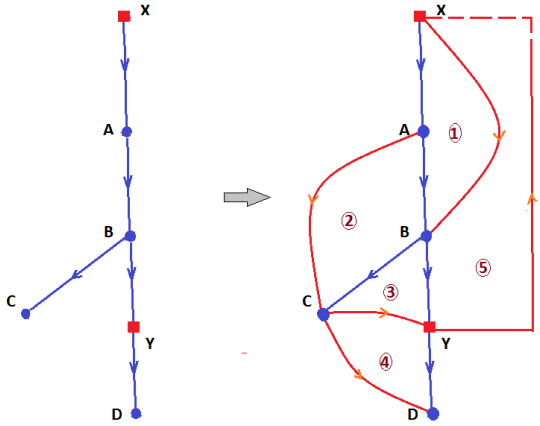

Fig. 8

Back edges are then fitted to construct fundamental cycles, in order to establish the condition equations. These cycles are: XABX, ABCA, BYCB, YDCY, and XBYX.

3. *Model Solution:*

A model solution arises out of comparison of the two search options, BFS and DFS. The consensus of opinion is that choice depends on the nature of data structure. In this regard, a survey route is characterized by depth and DFS is a natural representation. Furthermore, it does not entail huge memory requirements.

BFS nevertheless has advantage of minimum path length in cycle formation. Involving cross edges, the cycles have less overlap and agrees more readily with visual observations of geometry. However it does not lend with run of survey and topology.

Generally, as described in [5], the best option is offered by Depth - Limited Search, which is to say that thefinest solution lies in improving DFS. Thus, in this application, DFS is adopted and have proved very robust and reliable.

III. CONDITION EQUATIONS

Condition equations can be developed from the cycles formed in Fig. 8. Considering in the first instance a case of one-tuple, V = {H}, the equations are as follows:

$$H_{XA} + H_{AB} - H_{XB} = 0$$
$$H_{AB} + H_{BC} - H_{AC} = 0$$
$$H_{BY} - H_{CY} - H_{BC} = 0$$
$$H_{YD} - H_{CD} + H_{CY} = 0$$
$$H_{XB} + H_{BY} + H_{YX} = 0$$

where H_A is the MPV and $H_{XA} = H_A - H_X$. Continuing as in (1),

$$v_{H_{XA}} + v_{H_{AB}} - v_{H_{XB}} = C_1$$
$$v_{H_{AB}} + v_{H_{BC}} - v_{H_{AC}} = C_2$$
$$v_{H_{BY}} - v_{H_{CY}} - v_{H_{BC}} = C_3$$
$$v_{H_{YD}} - v_{H_{CD}} + v_{H_{CY}} = C_4$$
$$v_{H_{XB}} + v_{H_{BY}} + v_{H_{YX}} = C_5$$

In matrix format, $\mathbf{A}\boldsymbol{\vartheta} = \mathbf{b}$ and the expression is:

$$\underset{5\times 9}{\begin{pmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}} \underset{9\times 1}{\begin{pmatrix} v_{XA} \\ v_{AB} \\ v_{XB} \\ v_{BC} \\ v_{AC} \\ v_{BY} \\ v_{YD} \\ v_{CY} \\ v_{CD} \end{pmatrix}} = \underset{5\times 1}{\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}}$$

Fig. 9 Matrix Equation

It may be noted that in equation C_5, **YX** is a fixed line, and therefore a residual correction$v_{H_{YX}}$ is not applicable. Hence, this is reflected in the matrix equation.

A. N-Dimensional Network

Traversing is one of the most versatile and common means of establishing a control network [6]. Invariably, the mode of collection is automated and standard setting is usually three dimensions.

Computations of traverse survey always start with determining provisional positions. These are subject to further computations, an adjustment process for final results. Thus, irrespective of the mode of estimation, derived positions are available to check for internal and external consistency in geometry. And it would be more expedient to continue with determining the most probable values by least squares method of condition equations.

With reference to Fig. 8, a three dimensional network may be considered. It could be noted as in Fig. 1 that where there are sub stations, it would be more efficient to obtain the results in a secondary process by proportional error distribution. Hence this adjustment is in respect of primary nodes in the network, Fig. 6.

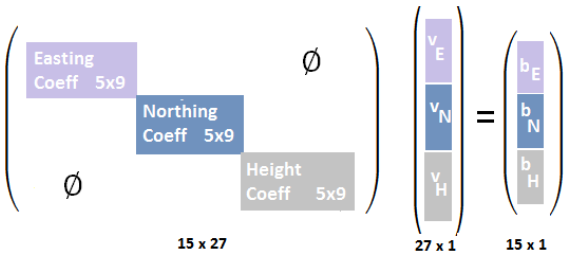

Fig. 10 3-D Matrix Equation

The cycles in Fig. 8, form the basis of a three dimensional condition equations. It involves constructing a matrix as in Fig. 9, with each representing a dimension. As in Fig. 10 therefore, there are three partitions for Easting, Northing and Height. Solution then follows the form of equation 3 and when it comes to covariance matrix, a weight scheme **W**, based inversely on number of sub stations is reliable. Error estimates may be obtained from $\mathbf{Q_{\hat{l}}} = \mathbf{Q_l}[\mathbf{I} - \mathbf{B^T}(\mathbf{BQ_lB^T})^{-1}\mathbf{BQ_l}]$ as in [7]. Thus networks of the form in Fig. 4 are ideally computable.

IV. PROGRAMMING

Programming in early days was based on the principle of information processing. The evolution today is knowledge processing embodying Artificial Intelligence. Thus, the current techniques require knowledge representation of human taught processes by method of engineering [8, 9].

A. Knowledge Representation

The purpose of knowledge representation is to encompass the descriptions of a network in a suitable variable. Since a digraph describes a survey network, that process requires graph data structures. Thus, a single program variable representation could hold all observations and description of a network.

In this context, while frames are a form of knowledge representation, object constructs are the implementation platforms [10, 11]. The start to this construction is with node stations and the edges that provide the path links.

1. *Node Object:*

```
PNodeRecord = ^TNodeRecord;
TNodeRecord = record
  StnName,StdErr: string[8];
                :
  InDegree,OutDegree,
  RefPath,FwdPath,OutPath:integer;
  PrvNodeHt,PrvNodeEt,PrvNodeNt,AdjNodeHt,
  AdjNodeEt,AdjNodeNt: string[15];
  AControl,Ticked,Marked:Boolean;
end;
```

```
PNodeObject = ^TNodeObject;
TNodeObject = object(TObject)
  NodeRecord: TNodeRecord;
  Angles: PCollection;
                :
  constructor Load(var S: TStream);
  procedure Store(var S: TStream); virtual;
  constructor Init(refStn:String;P:PCollection;
  refs,fwds,outs:integer;Ctrl:Boolean;Ht,Et,Nt:String);
end;
```

Fig. 11 Node Object

The node object, Fig. 11, starts with composition of record information. This record applies to every station in the network and thus covers requirements for sub stations, and results during computation.

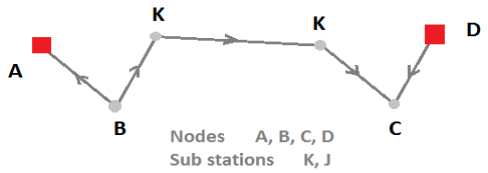

Fig. 12 Node Definition

A node may be described with respect to graph structure, as the station where the sum of In-degree or Out-degree is greater than one, Fig 12. The fields within the records therefore have this variable provision. Other complements include reference and forward paths for describing instances in a traverse network.

The status of nodes during various processing is important to avoid conflicts. This monitoring is accomplished with Boolean variables, Marked, Ticked etc.

2. *Edge Object:*

```
PEdgeRecord = ^TEdgeRecord;
TEdgeRecord = record
  FromStn, ToStn: string[8];
              :
  numOfSetUps, sign: integer;
  EdgeHtDiff,EdgeEtDiff,EdgeNtDiff,
  deltaHDiff,deltaEDiff,deltaNDiff,
  AdjHtDiff,AdjEtDiff,AdjNtDiff: string[15];
  Open,Ticked,Marked:Boolean;
end;
```

```
PEdge = ^TEdge;
TEdge = object(TObject)
  Stations: PCollection;
  EdgeRecord: TEdgeRecord;
                :
  constructor Load(var S: TStream);
  procedure Store(var S: TStream); virtual;
  constructor init(Points:PCollection;refNode,
  tgtNode: string; status:Boolean);
end;
```

Fig. 13 Edge Definition

The edge object follows the same pattern with node object by construction of record information. Important to note are the variables, FromStn and ToStn. They designate the direction of the edge in the diagraph, such as BA, BC and DC in Fig 12. Also of note is the number of setups that constitute the weights for each edge, useful in adjustment computations in the network.

Similarly, the Boolean variables, Open, Ticked, Marked and Status play key roles in monitoring paths that need to or have been processed during formation of a span tree. Equally, the sign variable serves the useful purpose of reversing observed direction during construction of fundamental cycles.

3. *Network Object:*

The network object is the next logical construction. This is in form of variable definition of a survey network, irrespective of size or dimension.

```
PNetWork = ^TNetWork;
TNetWork = object(TObject)
              :
  NodeList, Fixed :PCollection;
  EdgeList: PSortedEdges;
  StnNode:PNode;
              :
  numNodes,numControls:integer;
  Run:PEdge; List:PDataSet;
  CoordList:PCoordCollection;
  constructor Init(NodeCollection, Edgelist);
  destructor Done; virtual;
              :
  procedure FormNodes;
  procedure LinkNodes;
  procedure Add_ProvCoords;
              :
  procedure SaveData;
end;
```

Fig. 13 Network Definition

Of note are the variables, EdgeList and NodeList for storage mechanism. This is in preference, being more suitable to such options as Adjacency Matrix.

The methods FormNodes, LinkNodes provide a mapping description of the network. Add_PrvCoords provides preliminary positions and SaveData, saves the network to file for use in generating cycles and least squares computation.

B. Cycle Generation

In earlier discussions, the DFS was demonstrated as a more suitable algorithm for cycle processing. However, for implementation the mechanism differs from standard approach involving stacks etc. Rather what is adopted in this instance is analternative to suit survey applications.

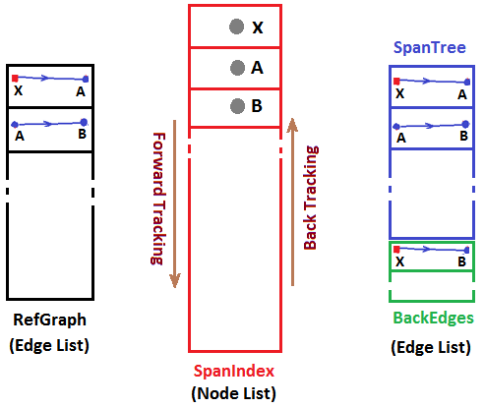

Fig. 14 Program Steps

With respect to Fig. 6 representing the network, a program implementation is described with illustrations in Fig 14. Thus, RefGraph is a variable representation for the structure as edge list. The same form applies to SpanTree and BackEdges. SpanIndex is a node list.

The process starts with initialization where SpanIndex, SpanTree are empty. The first segment XA, invariably the origin of the survey is evaluated and inserted into SpanTree. At the same time, the nodes X and A are inserted into SpanIndex. The target station A becomes the reference in forward tracking that yield the edge AB. Both the SpanIndex and SpanTree are subsequently updated. Forward tracking continues until a leaf, D, is encountered. Back tracking then follows until B is accessed and a new segment BC is found and explored, Fig 8.

Completion produces a DFS spanning tree. The back edges are generated, and form the basis for constructing fundamental cycles and condition equations.

```
PCycle = ^TCycle;
TCycle = object(TObject)
   DataFile: PResourceFile; Continue: Boolean;
                :
   Loop: PLoop; Loc: integer;
   EdgeCollection, NodeCollection,Ctrl,
   CycleRef, Segments,SpanIndex,
   refTree, Tree, cycleIndex : PCollection;
   SpanTree, refGraph :PSortedEdges;
   LoopFile:TResourceFile;
   LoopObjs: PSortedLoops;
                :
   constructor Init(DataFile: PResourceFile);
   destructor Done; virtual;
                :
   procedure InitializeData;
   procedure CreateASubTree;
   procedure BuildTheLoops;
   procedure TransferEdges;
end;
```

Fig. 15Cycle Object

The programming implementation of this construction can be achieved by Cycle object, Fig. 15. Of note are the fields that include refGraph, SpanIndex, SpanTree etc. The key methods are CreateASubTree and BuildTheLoops.

1. *Create A Sub Tree:*

```
procedureTCycle.CreateASubTree;
begin
New(SpanIndex, Init(500,50));
New(SpanTree, Init(500,50));
New(refGraph, Init(500,50));
TransferEdges;
FormSpanTree;  FindBackEdges;
AddCtrlEdges; SaveTree;
end;
```

Fig. 16

Create a sub tree begins with initialization to create SpanIndex, SpanTree and refGraph. Thereafter, TransferEdges builds a copy of the network in refGraph, an edge list structure. FormSpanTree then executes to create a SpanTree.

SpanTree in Fig. 17 starts after initialization, by constructing the first segment of the DFS spanning tree. Accordingly, reference and target nodes are registered in Node List. The outbound search for the next segment then begins with:

```
frNode := PString(SpanIndex^.At(num))^;
Key :=frNode;refEdge := refGraph^.FirstThat(@Search);
```

Operating in a repeat loop, the search moves in a forward direction, updating the Node List and SpanTree. When a leaf is encountered, back tracking commences with num := num -1.

At a stage the tracking returns to the root node from where forward search resumes for inbound observations. It executes inside a repeat loop untilsearch is exhausted and no further edges are detected. AllDone is then flagged true and the construction is complete.

```
procedure FormSpanTree;
var num,u:integer;  frNode, toNode, Key,
LKey, StnRef:   string; refEdge,
nextEdge :PTravEdge;  AllDone : Boolean;
begin
  InitializeIndex; num := SpanIndex^.count-1;
  AllDone := False;
  repeat
    frNode := PString(SpanIndex^.At(num))^;
    Key := frNode;
    refEdge := refGraph^.FirstThat(@Search);
    if refEdge <> nil then
    begin
      toNode := refEdge^.EdgeRecord.ToStn;
      refEdge^.EdgeRecord.Ticked    :=    True;
      LKey := toNode;
      if (SpanIndex^.FirstThat(@LookUp) = nil) then
      begin
        SpanIndex^.Insert(NewStr(toNode)); { Update Node List}
        SpanTree^.Insert(refEdge);  { Update Span Tree List }
      end;
      num := SpanIndex^.count-1;  { Forward Tracking }
    end
    else
    begin
      num := num - 1;  { Backward Tracking }
      if num < 0 then
      begin
        u := 0;        { Return to the root }
        repeat
          StnRef := PString(SpanIndex^.At(u))^;
          nextEdge:= EdgeCollection^.FirstThat(@NewNode);
          u := u + 1;  { Inbound Forward Tracking }
        until (nextEdge  <>    nil) or (u = SpanIndex^.count);
      if nextEdge <> nil then
      begin
        FrNode  :=  nextEdge^.EdgeRecord.FromStn;
        ToNode  :=  nextEdge^.EdgeRecord.ToStn;
        SpanIndex^.Insert(NewStr(FrNode));
        nextEdge^.EdgeRecord.Ticked := True;
        SpanTree^.Insert(nextEdge);
        num := SpanIndex^.Count-1;
      end
      else AllDone := True;
    end;
  end;
until ( AllDone);
end;
```

Fig. 17 Span Tree.

The routine concludes with FindBackEdges, constructing the back edges. This is derived from the main graph as missing links in the spanning tree. By expression: BackEdges = refGraph – SpanTree.

In the same measure, AddCtrlEdges adds the external links in the network to a common list database, CycleIndex.

2. *Build The Loops:*

```
procedureTCycle.BuildTheLoops;
begin
FetchBackEdges;
:
ConstructCycles; LoopClosure;
SaveCycles;
end;
```

Build the loops begins with fetching a tree description of the network as edge list structures.

Next a similar exercise obtains a list of back edges to enable construction of cycles to begin.

```
LoopCount := -1; g := 0;
cycleIndex :=
PCollection(LoopFile.Get(StrPas(cycleIndexRef)));
Tree  := PCollection(LoopFile.Get(StrPas(TreeIndex)));
repeat
   New(Segments,Init(50,50));
   ref := PTravEdge(cycleIndex^.At(g));
   New(R, Init(nil,",",False));
   R^ := ref^; Segments^.Insert(R); R^.Done;
  Trace;
   if onTarget then
   begin
     New(Loop,Init(Segments,de,dn,dh,accu,accu2));
     LoopObjs^.Insert(Loop);
     LoopCount := LoopCount + 1;
   end;
    g := g + 1;
 until g = cycleIndex^.count; Tree^.Done;
LoopFile.Done; cycleIndex^.Done;
```

Fig. 18

The process ConstructCycles, Fig. 18, starts with initialisation, creating a segment list for each cycle, and setting LoopCount and g to zero. The variable ref stores the first back edge, and that is followed by insertion into Segments list. Trace then searches for other segments to make up the cycle. If on Target is true, the search is successful and a Loop object is formed. It may be noted how closures de, dn, dh in the cycle and linear accuracy are determined during initialization. The routine operates in a repeat loop until all the back edges are used in forming fundamental cycles in the network.

Condition equations can then follow as described in Figs. 8, 9 and 10.

V. APPLICATION

The application of this algorithm has proved very successful [12] in level and traverse networks.

A. Level Network

There are two considerations for illustration, namely description of the network and cycles in the network. It involves a large field survey.

1. *Description:*

Description starts with the edges, runs in the survey network. Each run has a source and terminating node. Also the edges are weighted with such information as number of setups and height difference, Fig. 19. There are about 311 edges in the survey network.

The node details complement the edge description. There are 122 stations with in and out degrees at each station. The provisional heights complete the information to provide a picture of the network.

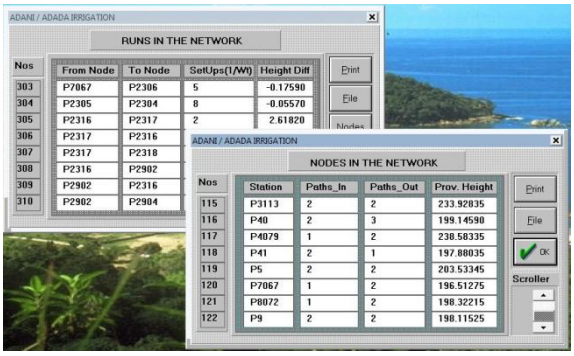

Fig. 19

2. *Cycles:*

Cycle formation is the next logical step that follows the descriptions. This is illustrated in fig. 20.

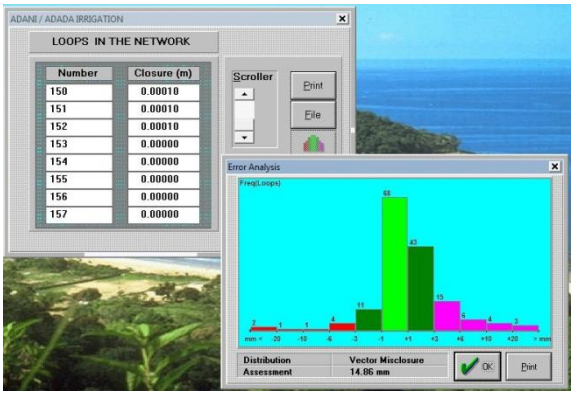

Fig. 20

It involves as many as 158 cycles and requiring same number of condition equations. A button lists the segments in each cycle while histogram is also available to show the spread of closures for analysis.

B. Traverse Network

Traverse network is also modelled on descriptions and cycles in the network. This illustration involves a 3-D survey comprising 15 stations.

1. *Description:*

As with level network, description is based on lists of runs, from a source node to a target node. The edge information also includes the number and list of instrument stations and 3-D coordinate differences in the network, Fig. 20.

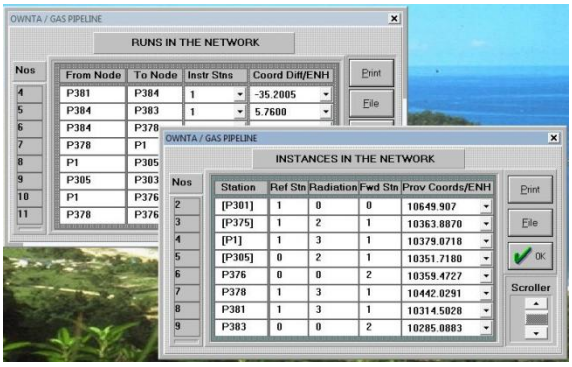

Fig. 20

Node details include instances as reference and forward stations in the network. The number of radiations is available to assist reconstruction of the network. Often field topology may differ from design and this facility is useful. The computed provisional coordinates are also available for determining closures in the survey.

1. *Cycles:*

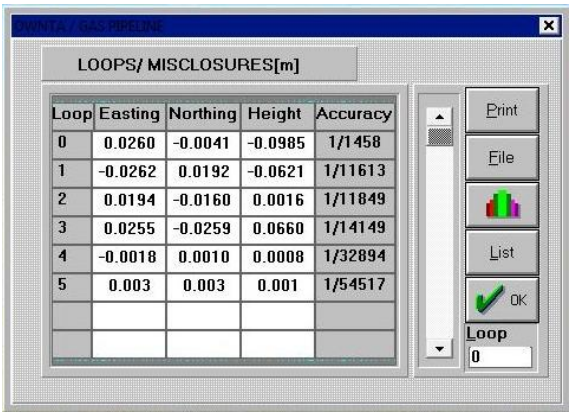

Fig. 21

There are six cycles in the network with closures and linear accuracy as illustrated in each case, Fig 21. A List button generates the segments in each cycle. The six cycles lead to formation of 18 condition equations for a 3-D network.

VI. CONCLUSION

It is generally accepted of the need for checks on consistency in networks prior to computations. Even in the much acclaimed parametric model of least squares adjustment, provisional coordinates are prerequisites to computations, such that an obvious complement is to profile resultant closures and linear accuracies in the network. Hence, the next logical step would be to incorporate cycle processing in the algorithm.

Survey instruments continue to evolve and in the current model of Total Stations, options exist for observations in coordinate mode. And if this option is hardly taken, it is because of issues of computability, because most software packages would rather process inputs involving direct observables, angles and distances. In this paper, the constraints have been removed. As such coordinate observations directly measured by Total Stations and GPS can be computed to the same standard as in precise level network. That is in n-dimensions, by least squares methodof condition equations.

Condition equations have always been preferred for safeguards that are inherent in use and memory efficiency. This paper has restored it for ascent, appreciation and as a ready complement in resolving survey networks.

REFERENCES

[1] M. A. R. Cooper,"Fundamentals of Survey Measurement and Analysis", Granada, 1982.

[2] Donald E. Knuth, "The Art of Computer Programming, Vol. 1 (3rd Edition): Fundamental Algorithms", Addison Wesley Longman Publishing Co.

[3] Paul Van Dooren, "Graph Theory and Applications", Université Catholique de Louvain, Belgium. Dublin, August 2009.

[4] Jean-Paul Tremblay, Paul G. Sorenson, "An Introduction to Data Structures With Applications", McGraw Hill Computer Science Series 2nd Edition.

[5] R. C. Chakraborty, "Problem Solving, Search and Control Strategies: AI Course Lecture 7-14",RC Chakraborty,2010, www.myreaders.info

[6] Yuji Murayama Surantha Dassanayake, "Fundamentals of Surveying Theory",Division of Spatial Information Science, University of Tsukuba.

[7] M. A. R. Cooper, "Control Surveys in Civil Engineering", Collins 1987.

[8] C. P. E. Agbachi, "Optimisation of Least Squares Algorithm: A Study of Frame Based Programming Techniques in Horizontal Networks", IJMTT, Volume 37 Number 3 - September 2016.

[9] C. P. E. Agbachi, "Design and Application of Concurrent Double Key Survey Data Structures", IJCTT, Volume 36 Number 3 - June 2016.

[10] J.E. Akin, "Object Oriented Programming Concepts", ©2001 J.E Akin, https://www.clear.rice.edu/../oop3.pdf

[11] Marco Cantu, "Object Pascal Handbook", © Marco Cantu 1995-2016. http://www.marcocantu.com/objectpascal

[12] C. P. E. Agbachi, "Surveying Software", Chartered Institution of Civil Engineering Surveyors ICES, October 2011, http://mag.digitalpc.co.uk/fvx/ces/1110/?pn=44