ITERATIVE APPROACH TO POSITION LOCATION IN PACKET RADIO NETWORKS

by

Julio Escobar

"B.Sc. in Electronics", The Victoria University of Manchester, England

July 1982

Submitted to the

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

in partial fulfillment of the requirements

FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1984

Signature of Author.. ....

Department of Electrical Engineering and Computer Science

December 19, 1984

Certified by... ....

P.A.Humblet, Thesis Supervisor

Accepted by...

A.C. Smith, Chairman, Committee on Graduate Students

ITERATIVE APPROACH TO POSITION LOCATION IN PACKET RADIO NETWORKS

by

Julio Escobar

Submitted to the
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
On December 19, 1984 in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

ABSTRACT

An iterative approach to the problem of reconstructing the relative positions of the nodes in a Packet Radio Network when some internode distance information is available (Position Location problem) is proposed.

The Position Location problem is expressed as a nonlinear minimization problem where configurations of the network compatible with the distance information available correspond to global minima of a suitable cost function.

The structure of the family of cost functions proposed is analysed and we establish that the cost functions, in certain instances, not only may not be globally convex but may have non-optimal local minima. A possible way to overcome this dificulty is suggested. Analysis of the Hessian matrix of the cost function reveals that this matrix is positive semidefinite at the global minima. Hence the cost functions proposed are locally convex at global minima points. Steps to make the cost functions strictly convex at global minima are described.

A software package implementing our iterative approach was developed and was applied to a few sample networks. Convergence to valid configurations of the networks was achieved in several instances. Instances of convergence to non-optimal local minima are also presented. Finally the effect of errors in distance measurements on the results obtained through the iterative approach are analyzed for the case where the errors are small.

Thesis Supervisor: P.A. Humblet
Title : Associate Professor of Electrical Engineering

A MITO Y TATI

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Chapter 2

Chapter 3

Chapter 4

LIST OF TABLES

CHAPTER 1

Chapter 1 gives an informal introduction to the subject of research. We explain the motivation behind it, review some relevant related work and give an outline of the thesis.

1.1 INTRODUCTION

In the field of computer communications packet-switched networks employing radios are of interest because they offer special attractive features, for example possible mobility of their elements. Networks of this type present in several respects problems and features particular to the broadcast nature of the communication system.

A Packet Radio Network (PRN) is a collection of geographically distributed packet radio units (PRUs) utilizing packet-switched communications in a broadcast mode of operation. Packet Radio Network applications are suited to networks where the users need to be mobile, for example civilian fleets in urban services (e.g. police) and military applications. The Advanced Research Projects Agency (ARPA) has set up an experimental PRN in California were new network software is continually being evaluated. A good review of PRN features can be found in [K et al].

One possible feature in Packet Radio Networks is the use of distance measurements for position location. Assume that any two PRUs in the network, that are within mutual range, can measure their separation, then distance information of this type among a set of PRUs could be employed by a processor to determine the relative position of the units involved. Typically not all units will be within range of each other, hence the distance information available will not correspond to all possible pairs of nodes in the network but rather to a subset of pairs. Besides, in situations where interference or physical obstacles are present, it may not be the case that any two nodes within range can measure their mutual distance. Thus, distance information available to the processor may be arbitrary.

A type of signalling waveform, which allows distance measurements, is Spread Spectrum modulation. Typical examples are Pseudo-noise (PN) modulation and Frequency Hopped (FH) modulation. Both systems employ a wide bandwidth for transmission. This wide bandwidth allows good time resolution for measurement of packet time of arrival which can be used to implement a Position Location System (PLS).

A position location system based on the type of information described, if feasible, could become a useful feature in PRNs, and a system of this type is the subject of our research. Position Location may be an end in itself: when location of nodes in the network is desired; or it could be exploited by algorithms taking advantage of position information to assist other functions, for example to decide on the best routing path for a stream of data packets. We have formulated and investigated the problem of position location having a Packet Radio Network scenario in mind. Position Location, however, is a problem interesting in its own right.

1.2 EXISTING POSITION LOCATION SYSTEMS

1.2.1 Fixed reference points available.

In general Position Location Systems use distance measurements to locate the position of one or more units of interest. When reference points of known positions are available and the measurements are with respect to these points, systems that exploit triangulation or similar techniques can be constructed. If no known reference points exist it is still possible sometimes, given enough measurements among the units of interest, to find the mutual relative positions of these units. This is the idea behind a PRN Position Location System.

The following 2-dimensional example illustrates one way in which location can be achieved if a set of reference points are available.

Consider 3 noncollinear points, P1,P2 and P3, whose positions are known, and a point R whose position is sought (see Figure 1.1-a). Let the distances from R to P1, P2, P3 (D1, D2, D3 respectively) be known. Let the position coordinates of P1 be $P1_x$, $P1_y$ and similarly for P2, P3 and R. This information then yields the following set of equations

$$(R_x-P1_x)^2 + (R_y-P1_y)^2 = D1^2 \qquad 1.1.a$$

$$(R_x-P2_x)^2 + (R_y-P2_y)^2 = D2^2 \qquad 1.1.b$$

$$(R_x-P3_x)^2 + (R_y-P3_y)^2 = D3^2 \qquad 1.1.c$$

corresponding to the loci of three circumferences of known centers and radii and with a common intersection point R. This system of nonlinear equations can be reduced to a system of simultaneous linear equations as follows:

Subtracting the equations pairwise (1.1.b from 1.1.a and 1.1.c from 1.1.a, say) and sorting yields

$$2(P2_x - P1_x)\ R_x + 2(P2_y - P1_y)\ R_y = K(P2,P1) \tag{1.2.a}$$

$$2(P3_x - P1_x)\ R_x + 2(P3_y - P1_y)\ R_y = K(P3,P1) \tag{1.2.b}$$

where K(Pj,Pi), a constant, is given by,

$$K(Pj,Pi) = Di^2 - Dj^2 + Pj_x{}^2 - Pi_x{}^2 + Pj_y{}^2 - Pi_y{}^2$$

Equations 1.2 constitute a system of inhomogeneous linear equations. The unknowns in the system are R_x and R_y. The system has a unique solution (the intersection of the circumferences) if the points P1, P2, P3 are not collinear, otherwise at least one of the equations 1.1 would be redundant and the system would have typically two solutions (Figure 1.1-b), one to either side of the line defined by P1, P2 and P3. Extension of this principle to n-dimensions is straightforward (c.f. section 4.5).

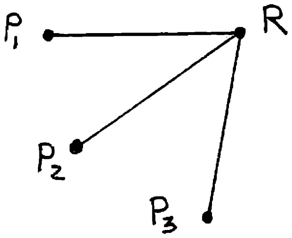


Figure 1.1-a

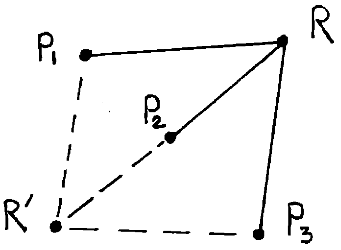


Figure 1.1-b

Most Position Location Systems use triangulation or some variant of it to calculate user's position. Examples are the Position Location Reporting System ([R],[KMM]), the Joint Tactical Information

Distribution System ([N],[GMR],[SS],[KS]), and the Acoustic Transponder Navigation System ([He]). This last system for example makes use of deep-sea acoustic transponders and is used by ships and submarines to determine their own position relative to the transponders.

The Navstar satellite system, for example, uses four satellites of known position and synchronized in time. The user receives from the satellites signals which contain position information and the signal time of departure. The user then calculates its distances to the satellites and sets up a system of four equations similar to 1.1. The equations now include three unknown space coordinates (the user's) and a fourth unknown which is the uncertainty in the distances calculated. This uncertainty is introduced by the possible lack of synchronization between the user and the satellites.

1.2.2 No fixed reference points available.

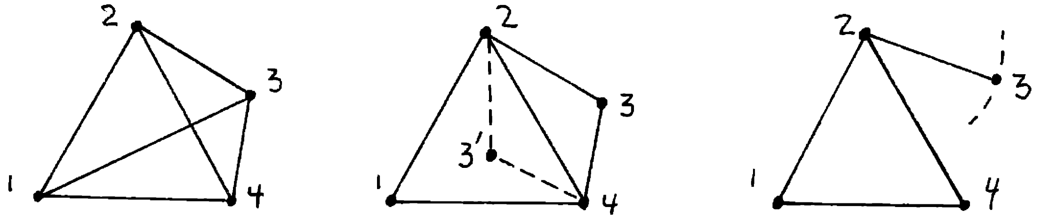
In a PRN scenario distance measurements need not be with respect to a fixed set of reference nodes. It might not be possible either to succesively apply triangulation, or some variant, and gradually expand a subset of nodes, acting as references, so that new nodes can triangulate to it. In some cases the solutions for the nodes involved might have to be found together for all of them (see for example Figure 1.4) and the systems above would fail. The information available results in a set of equations, similar in form to equations 1.1, but that cannot be reduced to a linear system in the way we obtained equations 1.2 and which is generally difficult to solve since most of the position coordinates in the equations are unknown. The problem however might still have a unique solution for the "relative positions" of the nodes even if a straightforward analytical solution of the equations is not present. Position Location Systems based on different principles are needed in these cases.

Work related to Position Location when reference points are not present is sparse. Most of it treats the 2-dimensional case because of its practical importance. The only work we are aware of having PRN applications in mind is due to Y. Yemini. His work is of particular importance since it treats theoretical aspects relevant to the problem and also proposes one algorithm for the problem, the Incremental Position Location System (IPLS). We are not aware of any other systems proposed in this field. We will outline IPLS below and refer the reader to [Y 81]. It will serve to illustrate present work and also as a point of contrast since our approach to the problem is different.

Before explaining IPLS let us comment on the nature of the problem.

Given a sparse set of distance measurements in a PRN, when is there enough information for the Position Location (PL) problem to have a solution?

We identify three situations regarding this solution: a solution does not exist, there is a countable number of solutions, or there are uncountably many solutions. For physical networks the first case corresponds to an inconsistent set of measurements (due to inaccuracy or mistakes for example). There is an important characterization in the framework of linear algebra to discriminate between the last two cases. The term "rigidity" is used to denote the case when the number of solutions is countable; it includes the case when there is a unique solution (Figure 1.2).



unique countably many (2) uncountably many

Figure 1.2: Examples of types of solutions for PL problems (2-dimensional example)

Yemini's IPLS [Y 81] treats the 2-dimensional case. IPLS searches through the distance information trying to identify subsets of points for which IPLS knows that their relative positions with respect to each other are unique and also knows how to find these relative positions. These subsets are denominated "hyperedges" and elementary examples are the triangle or a simple edge. With several hyperedges identified, those with common nodes are gradually merged using a set of "welding rules". If the result is a structure with unique solution it is considered a new hyperedge. If the result admits several (countably many) solutions the result is denominated a "superedge".

Conceptually, a superedge constitutes a "tree" of position knowledge, where branches represent possible relative configurations of the constituent hyperedges (Figure 1.3). A superedge needs extra information, in the form of measurements, hyperedges, or even other superedges, to become a hyperedge. Until such information becomes available the merging process must keep track of all possible configurations; all relative reflections of hyperedges as in Figure 1.3 for example must be allowed as possibilities. Once the necessary information arrives the correct combination of positions is chosen and the new hyperedge is formed. This process is repeated until all possible solutions are found.

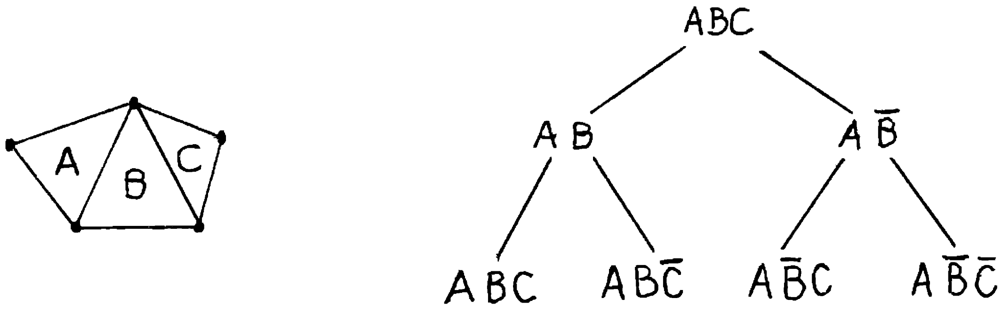


Figure 1.3: Example of "superedge" tree (2-d)
The set of hyperedges appears on the left. Capital letters label the hyperedges (triangles). Letters with an overbar in the tree denote a triangle that has been folded over the triangle immediately to the left of it.

In [Y 81] Yemini reports seven different welding rules; for example: two hyperedges with 3 noncollinear points in common can be merged into a single hyperedge. Using an analogy, welding rules take the place of equations 1.1 to gradually expand the rigid subset of points. The most complex rule in IPLS applies to the case when the topology shown in Figure 1.4 is found. The topology is identified by the algorithm and a numerical library routine is used to solve the equations for the positions. The computational cost however is high.

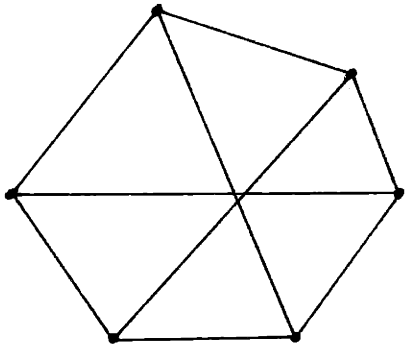


Figure 1.4: Example of welding rule

Lines represent edges or hyperedges. Points stand for nodes or sets of points depending on whether an edge or hyperedge is being treated.

As [Y 81] reports, this set of rules is not exhaustive; there are infinitely many "basic" rules,(i.e. rules which cannot be expressed as a combination of other rules) and the amount of computation required to solve the equations corresponding to basic rules involving more than, say, 10 nodes becomes too costly.

Some comments on IPLS are in order.

IPLS has important advantages:

- For the case with countably many solutions IPLS can determine in theory all possible solutions.

- Subsets of the PL problem are solved while the algorithm is in progress (e.g. assembling hyperedges), thus this knowledge is available early in the PL process.
- While in progress IPLS can, in theory, identify measurements needed to turn a superedge into a hyperedge. This could be used to place special measurement requests and speed up the PL process.
- Distributed versions of the system are possible, merging distributed knowledge afterwards using the available rules. For example, several processor nodes in a PRN could each apply IPLS to a local subset of nodes. The resulting local PL solutions would then be shared by the processors and merged at each of them using IPLS. Such an algorithm reduces the computational demand on the processor nodes since preprocessing takes place in a distributed fashion.

Some important disadvantages are:

- The set of basic welding rules is infinite. Even though most correspond to improbable configurations the possibility still exists that such configurations might be found. An extensive catalogue listing possible configurations and their solutions would be required and this could easily make the system impractical.
- Computation required for basic rules involving more than a few nodes is costly and may be too demanding in special configurations.

- Another drawback lies in the combinatorial nature of the algorithm. This is evidenced in part by the tree nature of the data representation of superedges.

Below we comment on this last issue, focusing on the graph search to identify hyperedges and on the problem of reflections.

IPLS searches through the graph underlying the given problem trying to identify subgraphs isomorphic to any of several graphs that IPLS maintains in its data base and whose solution is known. Searching for a given subgraph of c nodes in a graph G of V nodes can be done, if need be, by comparing all possible c-node combinations in G against the subgraph of interest. There are $V!/(V-c)!c!$ possible c-node combinations. This can be upperbounded by V^c. The comparison would take at most $c(c-1)/2$ operations (by considering all possible edges). This yields $o(c^2)$ operations. Thus an upper bound on the complexity of the graph search for a given item in the data base is c^2V^c, which is a polynomial growth on V. For a given subgraph this is a polynomial growth of complexity in the number of vertices of the graph G. However, since the maximum size of the subgraphs necessary in the data base grows without bound as a function of the number of vertices in G the search mechanism is in fact exponential in V, the number of vertices of the graph G.

The graph search and the use of the welding rules produce a set of hyperedges to be merged gradually. IPLS must keep track of possible configurations of these hyperedges until some new information determines the appropriate one. As illustrated in Figure 1.3 this implies an exponential growth of the operation count of IPLS as a function of the number of nodes in the network. In fact, the problem of finding the right combination of reflections to match the given information can be shown [Y 79] to be NP-complete. For every instance of the PARTITION problem (stated below) there is an equivalent instance of the problem of reflections.

PARTITION:

Given integers $C_1,\ldots,C_n$, is there a subset S of {1,2,...,n} such that

$$\mathrm{SUM}_{j \text{ in } S}[C_j] = \mathrm{SUM}_{j \text{ in } S^c}\ [C_j]$$

where S^c denotes the complement of S.

For the equivalent problem of reflections consider the following 1-dimensional network:

A- n nodes and n links.

B- The network is a cycle of n links.

C- The distance measurements are: $|C_i|,\ldots,|C_n|$.

Note: the order in which the distance measurements are assigned to the links does not matter (see below).

For a YES instance of the PARTITION problem the network defined has a valid 1-dimensional configuration. For a NO instance of the PARTITION problem the network defined does not have a valid 1-dimensional configuration.

Suppose we have a YES instance of PARTITION, let $S=\{N_1,\ldots,N_p\}$ and let $S^c=\{M_1,\ldots,M_q\}$. Let L_{Ni} denote the link corresponding to the distance $|N_i|$, and similarly define L_{Mi}. Define the sets $A= \{L_{N1},\ldots,L_{Np}\}$ and $A^c= \{L_{M1},\ldots,L_{Mq}\}$. Consider first the situation where the network consists of the cycle $L_{N1},\ldots,L_{Np},L_{Mq},\ldots,L_{M1}$. (Consecutive links share one common node.) Label V_0 the common node of L_{N1} and L_{M1} and label V_x the common node of L_{Np} and L_{Mq}. Starting from V_0 lay out the links L_{Ni} of A, orienting them in the positive or negative direction of the one dimensional space according to the sign of the integers N_i. Denote this path of links as path P. Repeat the process with A^c, also starting from V_0, and denote this path as path Q.

Since:

$$SUM_{i=1 \text{ to } p}[N_i] = SUM_{j=1 \text{ to } q}[N_j]$$

then, after laying out A and A^c as explained, we arrive at the same point through paths P and Q, namely at the position of V_x. We have then a valid one dimensional configuration for the network.

To show that the ordering of the links (i.e. the assignment of the distance measurements) in the cycle is irrelevant define a "Transfer" operation on link L_{Ni} (L_{Mi}) as follows:

Transfer of L_{Ni}:

Remove link L_{Ni} from the set A.

Remove integer N_i from the set S.

Relabel sets A and S as A_1 and S_1 respectively.

Bring together the end points of L_{Ni} in path P. (Hence the position of the last node in P is changed by an amount -Ni.)

Add link L_{Ni} to the set A^c.

Add integer $-N_i$ to the set S^c.

Relabel sets A^c and S^c as $A_1{}^c$ and $S_1{}^c$ respectively.

Insert link L_{Ni} in path Q (anywhere) so that it contributes an amount $-N_i$ to the position of the end node of Q.

In a Transfer operation we start with sets A, S, their complements, and a cycle network with paths P and Q of equal length (signed length). After the Transfer of link L_{Ni} we end up with sets A_1, S_1, their complements, and the paths P and Q for the resulting cycle still have equal length since they were both altered by the same amount. (Further, the lay out of the links in the resulting paths is also consistent with the integers in A_1, S_1, and their complements so S_1 and $S_1{}^c$ is a valid Partition for the problem.)

Thus, with Transfer operations we can turn the original cycle network into another cycle network with arbitrary link ordering and a valid 1-dimensional configuration would still be found.

Now, suppose we have a NO instance of PARTITION. If there were a valid 1-dimensional configuration for the network we could arrive at a valid Partition for the PARTITION problem (a contradiction) in the following way.

Choose any two vertices of the network as V_0 and V_x. Obtain the corresponding sets A_1, $A_1{}^c$, S_1, $S_1{}^c$, say, from the paths P and Q defined by V_0 and V_x. The union of the sets S_1 and $S_1{}^c$ contains integers whose magnitude is the same as in the set $\{C_1,\ldots,C_n\}$ but may possibly have the wrong sign for some of these integers. Using Transfer operations we can turn A_1, A_1, S_1,$S_1{}^c$ into A, A^c, S, S^c, such that all integers have the signs corresponding to the instance of PARTITION given. Since a Transfer operation will change the sign of the integer associated with the link Transferred and results in a valid configuration (for the new cycle network that is produced) we can always achieve the signs desired for any integer and have a valid configuration. And since paths P and Q will thus have equal length.

$$\mathrm{SUM}_S[C_j] = \mathrm{SUM}_{S^c}\ [C_j]$$

Sets S and S^c constitute a valid Partition contradicting the assumption that we have a NO instance.

Hence if we have a NO instance of PARTITION there cannot be a valid 1-dimensional configuration for the cycle network described.

Even though the problem of reflections is NP-complete we have tried to avoid some of the combinatorial features of IPLS by expressing the PL problem as a nonlinear optimization problem. In such an approach there is no need to maintain an initial set of items in a data base to implement a graph search. Also the iteration algorithm does not keep track of all possible configurations. The problem itself however remains NP-complete but the performance of the algorithm in typical situations might be improved. Such an approach would also have greater applicability since no equivalent to a set of welding rules is needed a priori. We seek then to study if an iterative method for the optimization problem is capable of obtaining a solution for instances of the PL problem provided one exists.

The equations involved in the problem are quadratic. We produce an arbitrary assignment of positions to the nodes in the network and define a cost function which compares distances in this assignment with the distance information available for the problem. An iterative descent (minimization) method is then used to converge to the solution.

Our research is principally centered on investigating the nature and characteristics of the cost function. We have also explored the effect that measurement errors on the distances have on the solution found through the minimization method.

Note that the results arrived at by a Position Location algorithm apply to the network as it was when the distance measurements were made. We have formulated the problem in this thesis in terms of a static network for simplicity. The results of an algorithm would also apply if the network configuration varies slowly with respect to the processing time of the algorithm. We have not attempted to analyse the effect of a network with mobile nodes.

1.3 THESIS OUTLINE

The present chapter is an introduction to the Position Location problem and a review of related work.

Chapter 2 defines most of the basic terms referred to in this thesis. It leads to the definition of "frame", which will model the information available for processing, and to that of "infinitesimal displacement" of a frame which is the basis for characterizing the "rigidity" of a frame. The possible nature of the solutions for a given problem are considered and a formal statement of the problem is given. The chapter also contains an overview of the results available relating the topology of the problem to the rigidity of the frame associated with it.

Chapter 3 defines two possible cost functions that could be assigned to the problem. Each gives some insight into the effect of the choice of cost function on the results derived. They have interesting properties in terms of local convexity at solutions (chapters 4 and 5). The expression for the gradient of the cost function is derived next. It leads to a simple physical analogy which helps to visualize the dynamics of the iterations and also suggests the existence of inflexion points other than the global minima. By construction we show that such inflexion points may exist and so the cost functions may not be globally convex. Considerations on the effect of these inflexion points on the descent method and possible ways to overcome this problem are also presented.

Chapter 4 contains the derivation of the Hessian matrix of the cost function (the matrix of second order partial derivatives). This matrix is necessary both for implementation of the descent method and, principally, for theoretical treatment of the local convexity properties of the cost function. The theoretical treatment of "rigidity" is presented here because of its relationship with the convexity of the cost function at solution points. Steps necessary to make the cost functions locally convex at the solutions, and their meaning, are discussed. If these steps are taken, the Hessian matrix at the global minima (the solutions) is found to be positive definite for well posed problems. We also state the descent method in full at this point since the relevant structures required have been derived.

Chapter 5 considers the effect of measurement errors (i.e. erroneous distance information) on the solution found through the descent method. We relate the errors induced in the positions found for the nodes to the errors in the distance measurements given. Both sets of errors are expressed as vectors and their relationship is strongly related to the Hessian matrix of the cost function chosen. The relationship is valid to a first order approximation and is derived by considering a Taylor's series expansion of the gradient around the modified solution of the problem. The relationship has an interesting interpretation in terms of linear algebra which corresponds to an intuitive view of the solution produced by the descent algorithm when measurement errors are present.

Chapter 6 contains a synthesis of the results in the thesis and the conclusions. It also includes suggestions for future research on the problem.

CHAPTER 2

In this chapter we give the basic definitions to be used in treating the Position Location problem. We then state the PL problem in a more rigorous way and give some considerations regarding the topology of the problem. For a reference on graph theory see [B].

2.1 PRELIMINARY

Some basic definitions follow below:

R^n: The n-dimensional vector space of real numbers.

$||r_i||$: The Euclidean norm of the vector r_i in a given vector space. We will denote by r_{ij} the vector difference $r_i - r_j$.

G(V,E): An undirected graph. V is the finite set of vertices of G(V,E). E is the finite set of edges of G(V,E). E={(i,j)} is a subset of VxV. (i,j) and (j,i) denote the same edge of E. G(V,E) has no self-loops, i.e. (i,i) is not an element of the set E.

Kv: Let v be the number of elements in the set of vertices V, then Kv is the graph G(V,E) where E is the set of edges containing all possible edges for V. In graph theory language Kv is the "complete graph" of v nodes.

d(i): The "degree number" of a vertex i. It equals the number of vertices adjacent to i. The degree number of a graph G(V,E), d(G), equals the maximum d(i) over the set of vertices V.

k(G): The "connectivity number" of a graph G(V,E). It equals the minimum number of vertices that need to be removed from G(V,E) to produce a disconnected graph or a trivial (isolated vertex) graph. (e.g. the connectivity number of the complete graph Kv is v-1.) A graph G is "n-connected" if $k(G) \geq n$.

$D(G)=\{D_{ij}$: (i,j) element of E$\}$: The "distance set vector" associated with the graph G(V,E). It has exactly one component corresponding to each edge of the graph. The components D_{ij} are positive real numbers. Two or more distinct components of D(G) may have the same value. Note that the components of D(G) are required to be non-zero. We sometimes refer to D(G) simply as the distance set.

$Q(V)=\{r_i\}$: A "realization" of the set of vertices V. $\{r_i\}$ is a set of n-dimensional position vectors, one for each vertex i of V in the R^n vector space. A realization of the graph G(V,E) is a realization of its set of vertices V. Let r_{ij} denote the vector r_i-r_j, we say that Q(V) "is contained in" Y dimensions if the set $\{r_{ij}$:(i,j) element of E$\}$ spans Y dimensions. Two realizations are equal if position vectors corresponding to equal vertices are equal in both realizations. A realization can be denoted by an augmented vector, the "realization vector", listing all independent variables in the system, namely the coordinates of the r_i position vectors of the realization. Because there is a one to one correspondence between this vector and the realization it represents we use the symbol Q(V) interchangeably.

$M_G(Q(V))$: The measurement function. It maps a realization vector

Q(V) onto a distance set vector associated with G(V,E) according to $D_{ij} = ||r_i - r_j||$.

$T_n(G,D(G))$: A "frame". If there exists a realization Q(V) which is contained in R^n such that $M_G(Q(V)) = D(G)$ we denominate the ensemble of G(V,E) and D(G) "a frame". Any realization Q(V) such that $M_G(Q(V)) = D(G)$ is termed "a compatible realization of" $T_n(G,D(G))$ (or simply "a realization of" $T_n(G,D(G))$).

Sometimes, when the meaning is clear, we suppress the subindex n from the symbol $T_n(G,D(G))$. The same applies to the arguments in D(G),Q(V),G(V,E) and T(G,D).

Fact 2.1: $T_n(G,D)$ has uncountably many realizations in R^n.

We can see this by noting that starting from any compatible realization contained in R^n any combination of the following operations with respect to an arbitrary reference system in R^n yields another compatible realization:

1- Translation.

2- Rotation.

3- Reflection (of all points in $T_n(G,D)$ through any given proper surface in R^n).

The Euclidean distance between any two points in R^n remains invariant under any of these operations [BR].

2.2 DISPLACEMENTS

The following definitions will help in the treatment of the problem. We follow in part and try to remain consistent with the literature (e.g. [L],[Y 79]).

Consider a graph G(V,E). Let t be a discrete or continuous variable. We define a "realization function" Q(V,t) of the variable t whose values are realizations of V. We will assume that V is implied in the realization and, for the definitions in this and the next section, we will write simply Q(t). A realization function Q(t) can be discrete or continuous. For simplicity in the definitions below, whenever we refer to a continuous realization function Q(t) it is assumed to be defined in the interval [0,T] of t, and when we refer to a discrete realization function Q(t) it is assumed to be defined at the two values of t given in the sequence {0,T}.

It will be useful to introduce the concept of Isometric Transformation. An "Isometric Transformation" of R^n is a function of the parameter t, discrete or continuous, that maps R^n onto itself and assigns, for every value of t, an image point p' to every point p in R^n such that for all pairs of points p and r, the Euclidean distance between the image points p' and r' equals the Euclidean distance between p and r.

Euclidean distances are invariant under Isometric Transformations. Translation and/or rotations are examples of Isometric Transformations. Isometric Transformations are one to one, otherwise Euclidean distances would not remain invariant for all pairs of points under the Isometric Transformation. Since Isometric Transformations are also onto they are invertible transformations.

def 2.1: A "displacement of T(G,D)" is a realization function Q(t) of t, continuous or discrete, such that

$$M_G(Q(t)) = D(G)$$

for all values of t for which the realization function is defined.

A displacement T(G,D) is a function whose values are realizations compatible with T(G,D). This function defines a trajectory (for the continuous case) or an assignment (for the discrete case) of positions for the vertices of the frame, and for all values of t in which it is defined the distances corresponding to every edge of the frame are preserved.

def 2.2: A "small displacement of T(G,D)" is a continuous differentiable realization function Q(t), defined in an interval [0,T], such that

$$M_G(Q(t)) = D(G) + o(t)$$

where the term o(t) is a term such that the limit of the ratio o(t)/t as t tends to zero equals zero.

Note that the values of a small displacement need not be compatible with T(G,D). Hence a small displacement is not a displacement proper.

def 2.3: A "trivial displacement of T(G,D)" is a displacement of T(G,D), discrete or continuous, which could be generated by an Isometric Transformation.

In a trivial displacement the trajectory (for the continuous case) or the assignment (for the discrete case) of positions for the vertices of the frame would be the same as for some Isometric Transformation. Definition 2.3 implies that the distance between any two vertices in a frame are invariant under trivial displacements of the frame.

Figure 2.1 illustrates examples of displacements.

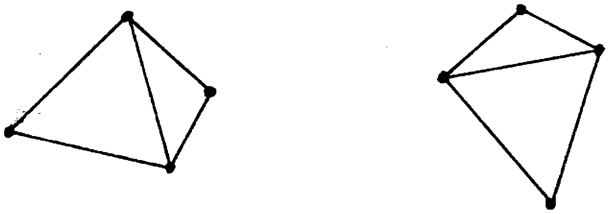


Figure 2.1-a: trivial displacement

(translation plus rotation)

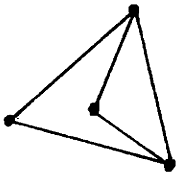


Figure 2.1-b: non-trivial displacement

(partial reflection)

def 2.4: An "infinitesimal displacement of $T(G,D)$" is defined to be the derivative with respect to t, evaluated at t=0, of a small displacement.

For infinitesimally small values of t the loci of the vertices in a small displacement tend to (infinitesimally small) vectors tangent to the initial trajectory of the vertices in the small displacement. Thus an infinitesimal displacement is equivalent to the assignment of a velocity vector to each vertex in the realization $Q(0)$. Note that an infinitesimal displacement is not a displacement.

def 2.5: Two realizations, Q_1 and Q_2, of $T(G,D)$ are said to be "equivalent" if either one can be obtained from the other by an Isometric Transformation.

Lemma 2.1: Let $Q_1(V)$ and $Q_2(V)$ be two realizations in n-dimensions of the graph $G(V,E)$. Let v be the number of vertices in $G(V,E)$. Then $Q_1(V)$ and $Q_2(V)$ are equivalent, for $T_n(G,D(G))$, if and only if

$$M_{Kv}(Q_1) = M_{Kv}(Q_2)$$

Note: K_v is the "complete graph" (defined in section 2.1). The equation simply means that the distances between each possible pair of vertices in one realization must equal the corresponding distances in the other realization for they to be equivalent.

Proof:

Necessity:

Distances are invariant under Isometric Transformations. From definition 2.4, if Q_1 and Q_2 are equivalent they are related by an Isometric Transformation. Hence the same measurement function applied to both realizations gives the same distance set and the equality becomes a necessity.

Sufficiency:

Let r_i denote the position vector of vertex i (i in V) for realization Q_1 and p_i the position vector for the same vertex i in realization Q_2. Let $\langle r_i, r_j \rangle$ = denote the usual inner product of vectors r_i and r_j in the finite dimensional vector space where Q_1 and Q_2 are defined. Choose any vertex k of V and consider the two sets of vectors G_1 and G_2 defined as:

$$G_1 = \{x_j = r_j - r_k : j \text{ in } V\}$$

$$G_2 = \{y_j = p_j - p_k : j \text{ in } V\}$$

Then G_1 and G_2 both contain a common vector, namely:

$$x_k = y_k = 0\text{-vector}$$

From the condition $M_{Kv}(Q_1) = M_{Kv}(Q_2)$ we have

$$||r_{ij}|| = ||p_{ij}||$$

$$\langle r_{ij}, r_{ij} \rangle = \langle p_{ij}, p_{ij} \rangle \qquad \text{for all } i,j \text{ in } V$$

by definition of $M_G(Q)$ and of the graph Kv (complete graph).

This implies that:

$$\langle x_{ij}, x_{ij} \rangle = \langle y_{ij}, y_{ij} \rangle \qquad \text{for all } i,j \text{ in } V$$

since $x_{ij} = r_{ij} = p_{ij} = y_{ij}$

Hence,

$$<x_i,x_i> - 2<x_i,x_j> + <x_j,x_j>$$
$$= <y_i,y_i> - 2<y_i,y_j> + <y_j,y_j>$$

for all i,j in V

but since $<x_i,x_i> = <x_{ik},x_{ik}>$

$$= <y_{ik},y_{ik}> = <y_j,y_j>$$

then $<x_i,x_j> = <y_i,y_j>$ for all i,j in V

Now, for two sequences of vectors $S_1=\{x_1,\dots,x_n\}$ and $S_2=\{y_1,\dots,y_n\}$ defined in the same inner product space , a necessary and sufficient condition for a linear Isometric Transformation U to exist, such that $U(x_i) = y_i$, i=1,...,n, is that the sequences S_1 and S_2 have the same "Gramian" [H]. The Gramian for a sequence of vectors $\{q_1,\dots,q_n\}$ is defined as the n by n matrix whose i,j entry is given by $<q_i,q_j>$, i,j=1,...,n.

Hence, our inner product conditions on sets G_1 and G_2 stated above are necessary and sufficient conditions for a linear Isometric Transformation U: $R^n \rightarrow R^n$ to exist such that $U(x_j) = y_j$.

But,

$$U(x_j) = U(r_j-r_k) = U(r_j)-U(r_k) = y_j = p_j-p_k$$

for all j in V. Hence a valid Isometric Transformation T: $R^n \rightarrow R^n$, non-linear in general, such that $T(r_j) = p_j$ for all j in V is:

$$T(r_j) \equiv U(r_j) - U(r_k) + p_k , \qquad k \text{ in } V$$

Hence, realization Q_2 can be obtained from Q_1 by the Isometric Transformation T and, from definition 2.4, they are equivalent.

def 2.6: A "Position", P_i, of a frame is a class (a set) of mutually equivalent realizations of the frame.

Clearly each realization of a frame must belong to at least one Position of that frame. Note that from lemma 2.1 each possible realization of a frame belongs to only one Position of that frame. Subscript i is used to identify different possible Positions of the same frame. The set of all Positions of a frame is denoted by P.

2.3 RIGIDITY

The definition in this section classify frames according to the nature of the displacements they admits. Again we have followed and tried to remain consistent with existing literature (e.g. [Y 79], [Y 81]).

def 2.7: A frame $T_n(G,D)$ is "rigid" if all its differentiable continuous displacements are trivial displacements.

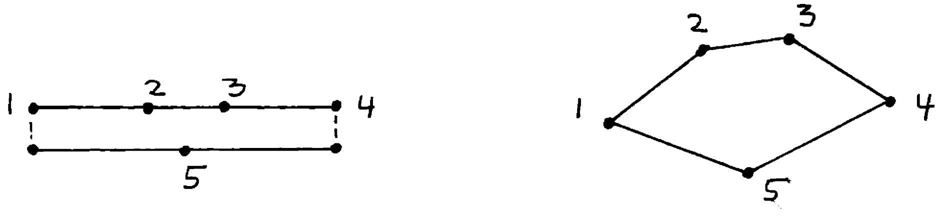
As a special case of rigid frame we have the following definition.

def 2.8: A frame $T_n(G,D)$ is "completely rigid" if all its displacements, discrete or continuous, are trivial displacements.

A completely rigid frame has a unique Position since all its displacements are trivial and hence obtainable from an Isometric Transformation. Thus all its realizations are mutually equivalent. Conversely, if a frame has a unique Position then it is completely rigid.

def 2.9: A frame $T_n(G,D)$ is "loose" if it admits non-trivial differentiable continuous displacements.

Note that the dimension n of T(G,D) should be specified when refering to its rigidity and possible Positions. It could be the case that from a given pair G(V,E) , D(G) we can obtain a rigid frame if n-dimensional but a loose frame if (n+1)-dimensional. Consider as an example the cycle graph of Figure 2.2 in 1 and 2 dimensions as shown.



cycle in 1-D

cycle in 2-D

Figure 2.2: rigid frame in n-dimensions and loose frame (n+1)

(same underlying graph and distance vector) def 2.10: A frame $T_n(G,D)$ is "infinitesimally rigid" if for all its infinitesimal displacements the associated small displacements are trivial displacements.

The property in definition 2.10 is particularly important. It is a local property and very relevant to an important characterization of rigidity. We treat this topic in chapter 4 since it is closely related to the analysis of the descent algorithm under study.

Note that, for a given dimension, infinitesimal rigidity of a frame also implies rigidity, as follows.

All small displacements are continuous and differentiable (definition 2.2). To each small displacement there is an associated infinitesimal displacement (definition 2.4). From definition 2.10, if a frame is infinitesimally rigid all its small displacements are trivial displacements.

Since any differentiable continuous displacement is a small displacement in the interval [0,T] (with o(t)=0), then all differentiable continuous displacements of the frame are trivial displacements. Hence, by definition 2.7, the frame is rigid.

We can extend the notion of rigidity. Because to every realization Q(V) of a graph G(V,E) there corresponds exactly one frame (whose distance set is specified by $M_G(Q)$) definitions 2.7 to 2.10 can also be applied to realizations of a graph. For example, we can say that a realization of some graph is rigid if the corresponding frame is rigid.

2.4 MODEL AND PROBLEM STATEMENT

From the definition of Position (definition 2.6) we see that the reference system used to express the realizations is irrelevant. This captures the nature of the PRN Position Location problem.

Consider a PRN in n-dimensions. A distance measurement process generates, conceptually, a graph G(V,E), whose edges stand for the existence of a measurement, and a distance set vector D(G) specifying the measured distances. Assuming exact measurements and since the PRN exists in Euclidean space, there will be at least one class of realizations compatible with D(G) and hence the information generated does constitute a frame $T_n(G,D)$.

The measurement process corresponds to a measurement function $M_G(Q)$, where G specifies which measurements are present and Q is the realization vector specifying the position of the nodes in the network. The value of $M_G(Q)$ is the distance set vector D(G) and the process results in a frame $T_n(G,D)$.

If measurements are inexact a modified distance set D'(G) is produced and two situations are possible: There exist some realization compatible with D', in which case the result of the process is a modified frame, or else there exist no realization compatible with D'. In this last case we refer to the information produced as a "pseudo-frame" since there is no Position for it in Euclidean Space.

The effect of measurement errors is considered in chapter 5. We will state the problem here assuming exact measurements. First we state a strict version of the problem.

The strict PL problem could be stated as follows:

INPUT: A frame $T_n(G,D)$.

QUESTION 1: Is $T_n(G,D)$ completely rigid ?.

QUESTION 2: If it is find a realization of $T_n(G,D)$.

If the answer to QUESTION 1 is YES and the realization found for QUESTION 2 is Q_0 then we are able to find the distance between any two nodes of the network since complete rigidity implies that all realizations of the frame are equivalent and from lemma 2.1, $M_{Kv}(Q_0)=M_{Kv}(Q)$, where Q specifies the positions of the nodes in the network. This capability would be a desirable feature in PRNs.

We can relax the problem statement for greater generality. A relaxed PL problem could be stated as follows:

INPUT: A frame $T_n(G,D)$.

QUESTION 1: Is $T_n(G,D)$ rigid ?.

QUESTION 2: If it is find a realization for each of the possible Positions.

The relaxed version of the problem is useful since a rigid frame which is not completely rigid may still represent valuable position information.

2.5 TOPOLOGY

Deciding for the rigidity of a given frame is of relevance to the PL problem. As would be expected the topology of the graph underlying the problem is closely related to this property. Some trivial examples: if the underlying graph is disconnected the frame is loose; if the underlying graph corresponds to a cycle of more than three edges the frame is loose for two or higher dimensions; if the underlying graph corresponds to a triangle the frame is completely rigid.

In fact it is easy to show that all frames whose underlying graph is complete are completely rigid:

Let realizations Q_i and Q_j be any two realizations of $T_n(G,D)$.

If G is a complete graph then,

$$M_G(Q_i) = D(G) = M_G(Q_j)$$

which by lemma 2.1 implies that Q_i is equivalent to Q_j and so $T_n(G,D)$ has a unique Position. Hence it is completely rigid.

Topology however is not enough to characterize rigidity in general. Figures 2.3-a and 2.3-b show two realizations for 2-dimensional frames of equal topology but different distance sets (note that one edge is hidden in Figure 2.3-b). The first case corresponds to a rigid frame with two possible Positions and the second to a completely rigid frame. In this section we state more precisely the role of topology on the problem by presenting some of the most important results available in the literature regarding the relationship between topology and rigidity.

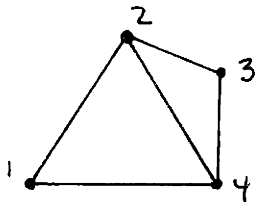


Figure 2.3-a

Rigid, 2 Positions

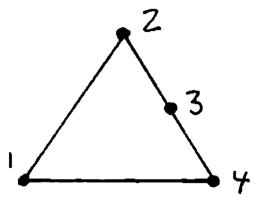


Figure 2.3-b

Rigid, 1 Position

(both frames have equal topology; one edge is hidden in 2.3-b)

To express the role of topology we will require the concept of connectivity (section 2.1) and theorem 2.1 (from Gluck [Gl]). His theorem was originally stated for 3 dimensions but can be generalized to n dimensions [AR]. We have adapted it to our definitions. This theorem will be discussed in greater detail in chapter 4 (see also [Y 81]).

Theorem 2.1: Let G(V,E) be a graph of v vertices. Let the position vectors of the vertices in a realization of G(V,E) be R^n, and identify the space of all possible realization vectors with R^{nv}.

Then if there is a frame $T_n(G,D_0)$ which is infinitesimally rigid, the set of frames T(G,D) (with D=D(G) arbitrary), of n dimensions or less, which are infinitesimally rigid corresponds to an open and dense set of realization vectors in R^{nv}.

Proof: See section 4.3

Theorem 2.1 implies that if there is a frame for G which is infinitesimally rigid then those frames of G which are not infinitesimally rigid correspond to realization vectors in R^{nv} belonging to a zero probability measure set with respect to a Borel probability measure absolutely continuous to the Lebesgue measure.

A topological characterization of infinitesimal rigidity is thus useful and meaningful. Specially since infinitesimal rigidity implies rigidity. We can speak of graphs with "rigid topology" meaning a graph for which loose frames correspond to degenerate situations in the sense of Theorem 2.1. The information available in such degenerate cases is simply not enough for the problem to have a solution. Figures 2.4-c illustrates one such situation. The explanation to Figures 2.4 is at the end of this section but for clarity we have referred to them here. The discussion below refers then to the typical case, i.e. excluding degenerate situations.

def 2.11: A graph is n-rigid if n-dimensional frames corresponding to this graph are infinitesimally rigid with probability one in the sense of theorem 2.1.

A similar reference could be made to looseness of a graph.

Because of its practical importance (and tractability), existing literature in the subject treats almost exclusively the problem of infinitesimal rigidity in the 2-dimensional (2-d) case. The problem of complete rigidity has received little or no coverage. We present first an overview of the 2-d case, trying to include complete rigidity, and then comment on the n-d case.

2.5.1 On complete rigidity.

The definition below will be useful. Let V be a set of vertices and Q(V) be a realization of these vertices in a real m-vector space.

def 2.12: A set V_{n+1} of n+1 vertices, subset of V, is "Maximally Spanning" if the set $Z_0 = \{r_{ij}: i,j \text{ in } V_{n+1}\}$ spans n dimensions.

Note: The definition arises from the following consideration. For any set V_{n+1} of n+1 vertices, the corresponding set Z_0 (as defined above) can span at most n-dimensions. To see this consider any node i of V_{n+1} and the set of vectors $Z_1=\{r_{ji}: j \text{ in } V_{n+1}, j \neq i\}$. Any vector r_{kl} of Z_0 can be expressed as a linear combination in the set Z_1 (e.g. $r_{kl}= r_{ki}-r_{li}$). Hence the span of Z_0 equals the span of Z_1. Since Z_1 has n vectors then the span of Z_1 can be at most n-dimensional. Hence Z_0 can span at most n-dimensions.

Finally, there will be instances of sets V_{n+1} for which the corresponding set Z_0 will in fact achieve the maximal span of n-dimensions; e.g. consider a situation, with $n \leq m$ (m as defined above), where the set V_{n+1} consists of a vertex at the origin and each of the remaining n vertices lie in a different orthogonal axis. (For $n>m$ it is not possible for a set V_{n+1} to be Maximally since the space of realizations itself is only m-dimensional.)

Also note that if a set V_{n+1} is Maximally Spanning then any set V_{k+1}, subset of V_{n+1}, is also Maximally Spanning. To see this assume that V_{k+1} is not Maximally Spanning. Then its corresponding set $Z_0=\{r_{ij}$: i,j in $V_{k+1}\}$ could span at most k-1 dimensions (or else it would be Maximally Spanning). If we add to the set V_{k+1} the remaining n-k vertices of V_{n+1} not already in V_{k+1} then the resulting set of vertices (V_{n+1} itself) can span at most n-1 dimensions since the added vertices could add at most n-k dimensions to the span of V_{k+1} (resulting in a total of n-1 dimensions at most).

<u>lemma 2.3</u>: V_{n+1}, subset of V, is Maximally Spanning if and only if <u>for any vertex</u> i in V_{n+1} the set of vectors $Z_2=\{r_{ji}$: j in $V_{n+1}\}$ spans n dimensions.

Proof:

Necessity: Suppose V_{n+1} is Maximally Spanning. Then the set $Z_0=\{r_{ji}$: i,j in $V_{n+1}\}$ spans n dimensions (from definition 2.12).

Take any vertex i of V_{n+1}, the set $Z_1=\{r_{ji}$: j≠i, j in $V_{n+1}\}$ is a basis for the affine space defined by V_{n+1} since Z_1 is a set of n vectors and all other vectors in Z_0 are linear combinations of these. If Z_1 is a basis then Z_2 spans n dimensions since Z_1 is a subset of Z_2.

Sufficiency: If Z_2 spans n dimensions so does Z_0 (since vectors in Z_0 are also linear combinations of Z_2), which means that V_{n+1} is Maximally Spanning.

lemma 2.4: Let G(V,E) be a graph of v nodes. Let Q(V) be a realization of G(V,E) contained in n-dimensions, $n \leq v$. Assume that any set of n+1 vertices in Q(V) is Maximally Spanning.

Then a necessary condition for Q(V) to be completely rigid in R^n is

$$k(G) \geq n+1$$

Proof:

Let k=k(G) and assume k<n+1. Then, by definition, there exists a set V_k (a "point-cut-set") of k vertices whose removal results in a disconnected graph (the removal cannot result in a trivial graph since v>n+1).

There must then be at least two subsets, V_1 and V_2, such that the three sets V_1, V_k, V_2 are mutually disjoint.

Because of the Maximally Spanning assumption in the lemma, V_k must be Maximally Spanning and hence defines an affine (k-1)-dimensional space, S_{k-1} say.

By the same assumption, neither V_1 nor V_2 can have vertices in S_{k-1}, otherwise there would exist a set of k+1 vertices which is not Maximally Spanning.

Hence, either V_1 or V_2 could undergo a proper reflection through S_{k-1} which would not correspond to an Isometric Transformation for the full graph but which would produce a new realization compatible with the distance set vector. Thus the frame associated with Q(V) would have more than one Position, i.e. Q(V) would not be completely rigid.

Lemma 2.4 can be made more general if we only required to be Maximally Spanning sets those sets of k+1 vertices containing some point-cut-set V_k as subsets.

For two dimensions 3-connectivity is a necessary condition for complete rigidity for graphs of four or more vertices. By inspection, graphs of less than four vertices must be complete to be completely rigid in 2-d (otherwise they are loose); this is also a sufficient condition. An example of a 3-connected graph whose realizations are typically completely rigid is that of Figure 2.4.

2.5.2 On sufficient conditions for rigidity.

A strong result presented by Lovasz and Yemini [LY] exists for the 2-d case.

Theorem 2.2: Every 6-connected graph is 2-rigid. Also, if three edges are deleted from a 6-connected graph the resulting graph is also 2-rigid.

Proof: see [LY]

The definition for 2-rigid graph is that of definition 2.11. Hence note that rigidity here is in the sense of theorem 2.1.

The proof of theorem 2.2 is based on important results derived by Laman [L] and because of their relevance we present them below.

v denotes the number of vertices of graph G

e denotes the number of edges in the graph G.

$v^{'}, e^{'}$ are defined similarly for subgraph $G^{'}$ of G.

Theorem 2.3: Every rigid 2-d frame $T_2(G,D)$ contains another subframe $T_2(G^{'},D^{'})$, also rigid, with v vertices and exactly 2v-3 edges, ($G^{'}$ subgraph of G, $D^{'}$ subset of D).

Note that 2v is the number of independent variables in the system and that 3 is equal to the number of degrees of freedom of an object in 2 dimensions.

Theorem 2.4: Any rigid 2-d frame $T_2(G,D)$ with v vertices and exactly 2v-3 edges has the following property:

If $G^{'}(V^{'},E^{'})$ is a subgraph of G(V,E) with $E^{'}=\{(i,j):\ (i,j)$ in E and i,j in $V^{'}\}$ and $v^{'}\geq 2$ then $e^{'} \leq 2v^{'}-3$.

Laman also shows that the conditions expressed by theorems 2.3 and 2.4 are sufficient conditions, in the sense of theorem 2.1, for 2-rigidity of a frame in two dimensions.

Theorem 2.3 says that every rigid frame (2-d) has a rigid "core" (rigid subframe), with the same number of vertices as the original frame, and that edges outside this core are not necessary for rigidity. Theorem 2.4 implies that such a core has no subframe which is rigid (since no subframe satisfies theorem 2.3), thus all edges in the core are needed to make it rigid; there are no redundant edges.

The proof of Theorem 2.2 uses these necessary and sufficient conditions. Lovasz and Yemini also show that 6 is the lowest sufficient connectivity number for 2-rigidity by constructing a class of 5-connected graphs which can be shown to violate Laman's conditions.

Regarding graphs of 6 or less nodes (which implies 5 or less connectivity) it is conceivable, in the abscence of other characterizations, to exhaustively test for 2-rigidity of topologies suspected of being rigid.

2.5.3 On extension of rigidity conditions to n-dimensions.

It is clear that 1-connectivity implies rigidity in 1-d problems. Despite efforts, a characterization of rigidity similar to Laman's theorems for the 2-d case has not been found for the n-d case, and the equivalent of Laman's result does not hold in 3-d [Y 81]. This makes the search of sufficiency conditions for rigidity in 3-d even harder. We are not aware of a topological characterization of rigidity for a general number of dimensions.

2.5.4 Testing for connectivity:

The importance of connectivity to the PL problem is evidenced by Theorem 2.2. Besides, connectivity of a graph can be tested for with polynomial time algorithms. These algorithms use the fact that the connectivity number of a graph is equal to the maximum number of point-disjoint paths (i.e. no common vertices) between "any" two vertices in the graph ([Ha],[B]). Application of the maximum flow algorithm can then be used to find this maximum number of disjoint paths. Even's algorithm [T] is a particularly efficient one.

A connectivity test for rigidity would answer incorrectly if the frame being tested has a rigid underlying graph but corresponds to a degenerate case. A typical example is illustrated in Figure 2.4. The example is shown for 2 dimensions. Figure 2.4-a just shows the topology of the graph underlying the example. Figure 2.4-b shows a frame with this topology which is rigid [L]. Figure 2.4-c shows a degenerate situation for the topology, the perimeter of the frame (Position A) corresponds to a regular hexagon; in this instance the frame is in fact loose [L]. [L] gives the analytic function,for the loci of the vertices, to move from Position A to B (Figure 2.4-c).

Note the abscence of triangles in the topology. A system using triangulation for example would not be suitable for this problem.

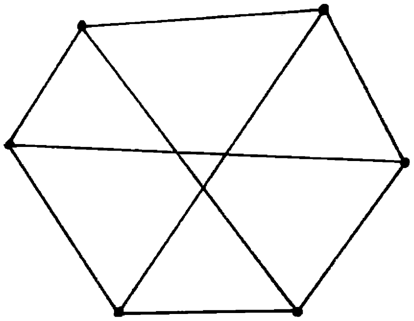


Figure 2.4-a: topology

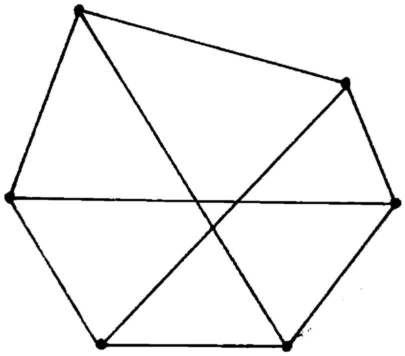
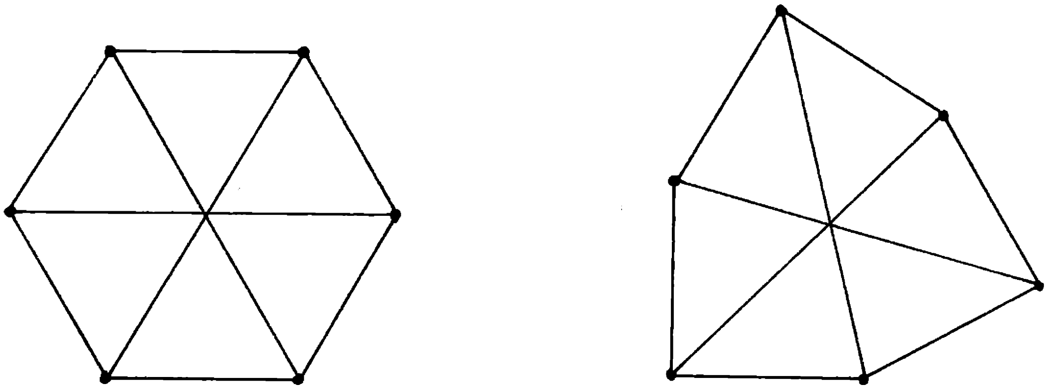


Figure 2.4-b: realization of a rigid frame



Position A

Position B

Figure 2.4-c: realization of a loose frame

CHAPTER 3

Having explained and formally stated the problem we now propose an approach for its solution.

Chapter 2 gave an overview of the tools available to answer Question 1 of the PL problem: What is the nature of the solutions ? (i.e. regarding rigidity.) We mentioned topological characterizations for rigidity. Results available cover mainly the one and two dimensional cases. In chapter 4 we will explain another characterization of rigidity through the idea of "infinitesimal rigidity" (definition 2.9).

In chapters 3 and 4 we address Question 2 of the PL problem: finding a solution when it exists. We will assume then that the input frame is rigid. Chapter 1 mentioned the Incremental Position Location System. We propose a different approach. Esentially we consider the use of an iteration method to find the answer. A suitable cost function will be defined and a descent method used to converge to the solution. We investigate this method by analysing the nature of two suitable cost functions.

Using a descent algorithm we expect to avoid in part some of the combinatorial aspects of algorithms like IPLS and the complexity of an analytic approach to the solution of the simultaneous nonlinear (quadratic) equations inherent to the problem. The problem remains NP-complete as mentioned in section 1.2.2 but we avoid the explicit graph search and keeping track of possible reflections which are present in IPLS.

3.1 SPACE OF OPERATION FOR DESCENT METHOD

We consider the frame $T_n(G,D)$ with graph $G(V,E)$ and distance set $D(G)$. We will first explain some of the concepts to be used in dealing with the PL problem and at the end of this section we include an example which will hopefully clarify these ideas.

Our descent algorithm starts from an arbitrary initial realization of G. This realization is represented by the realization vector Q_0. The independent variables consist of the vertex coordinates. The algorithm uses the first and second order derivatives of a cost function defined on these variables (section 3.2) to iterate towards a realization compatible with T. Several methods of iteration exist for treating nonlinear optimization problems. Simply stated, the kth step of iteration finds a realization Q_k using the equation

$$Q_k = Q_{k-1} + f(Q_{k-1}) \qquad 3.1$$

where f(.) is a function specified by the method chosen (see section 4.8).

For a frame $T_n(G,D)$ we will allow, in the algorithm, realizations of G in m-dimensions, with $m \geq n$. Thus we define the "space of operation" of the algorithm to be $S^+=R^m$, $m \geq n$. Conceptually, we have an arbitrary frame in S^+, whose underlying graph is also G, and which is gradually modified through equation 3.1 until a close approximation to the frame T is obtained.

Since the Positions of T are contained in R^n we find it useful to define a "space of solutions", S_0^+, as the subspace R^n of S^+, i.e. $S_0^+=R^n$. If the frame is rigid in n and higher dimensions we can take special provisions to make the possible solutions of the problem to be contained in S_0^+. These provisions are discussed in section 4.5.

Note that rigidity of a frame may depend on the dimensions of the space in which it is imbedded; Figure 2.2 showed an example of this. Hence if the input frame T, which is rigid in n-dimensions by assumption, is loose in the space of operation further provisions are necessary to converge to S_0^+ so that the solution for the problem is found. Again we defer mention of this topic to section 4.7.

Let m be the dimension of S^+ and v be the number of vertices in G. Then, from the analysis point of view, a realization of G in S^+ is an mv-vector Q. The components of Q are all the independent variables in the system, namely each coordinate of each vertex in G. Hence we define the vector space $S=R^{mv}$. It is in this space S that the cost function is defined.

A given realization of G can be thought of as some frame in the space of operations S^+ or as a point in the space S. We call S the "function space". Finally, the nv-subspace of S corresponding to the space of solutions is denoted by S_0.

The simple example that follows helps to illustrate the definitions given above.

Consider a 3-node network (PRN) in two dimensions and assume that the nodes are not collinear. A measurement process finds the distances between all pairs of nodes and the information is presented to an algorithm for solution:

The graph G(V,E) underlying the problem is a triangle.

The position of the nodes may be considered as a realization, Q_A say, contained in 2-dimensions.

The distances obtained through the measurement process are given by the vector $D_A= M_G(Q_A)$.

The input to the algorithm is then the frame $T_2(G,D_A)$.

The set P of Positions of T consists of 2-d realizations.

For a triangle there exists a unique Position. The Position Location Algorithm may then operate conceptually in a 3-d space of operation $S^+=R^3$, assigning 3-d position vectors to the vertices in G. With special provisions the iterations will converge to a realization compatible with T and contained in the two dimensional subspace $S_0{}^+=R^2$.

From the analysis point of view the algorithm starts with a 9-d realization vector Q_0 in the function space $S=R^9$. It uses the cost function and equation 3.1 to iterate to a realization vector Q_{end} in the space of solutions S_0. S_0 is a 6-d subspace of S where the solution is found.

Convergence of iteration methods in nonlinear optimization depends on the nature of the cost function being used, particularly on the behavior of the matrices of first and second order derivatives of the cost function, i.e. the gradient and Hessian matrices respectively. The analysis of these matrices in our problem is a major part of our research.

3.2 COST FUNCTION

Two choices of cost functions will be treated in this section. They correspond to a natural way of comparing an arbitrary realization of a graph with a given frame for that graph. Moreover, these cost functions, as will be seen, lead to interesting structures for their first and second order derivatives. Each of these cost functions leads to similar analytical results but it is useful to present their particular characteristics. Developing both of them serves to illustrate what is fundamental to the problem and what is particular to each of the cost functions.

Consider the frame $T_n(G,D)$. Let Q denote a realization vector of G and P denote the set of Positions of T. Assume that T is rigid.

For a given realization vector Q we consider cost functions C(Q) of type A and B (3.3.a and 3.3.b respectively). For the purpose of explanation we will refer to the following terms when dealing with the cost functions:

$C^A(Q)$: Type A cost function. Similarly for type B.

r_{ij}: The "edge vector" corresponding to edge (i,j), obtained from the realization by $r_{ij}= r_i-r_j$.

u_{ij}: The unit vector pointing in the direction r_{ij}.

e_{ij}: The "edge error" corresponding to edge (i,j). It is particular to each type of cost function.

A_{ij}: The "edge weight" corresponding to edge (i,j) for type A cost function ($A_{ij}>0$). Similarly for type B.

p_A: A normalization constant for type A cost function ($p_A>0$). Similarly for type B.

In our definitions, a cost function C(Q) is a real scalar function defined on a real vector space. It is a map C: S-->R, where S is the function space and R is the field of real numbers.

Below we define general versions for three possible choices of cost functions. E is the set of edges of G and D_{ij} are the distance measurements given.

$$e^A_{ij} = ||r_{ij}||^2 - D_{ij}^2 \qquad 3.2.a$$

$$C^A(Q) = (1/p_A)\ \mathrm{SUM}_E[\ A_{ij}(e^A_{ij})^2\ ] \qquad 3.3.a$$

with p_A and A_{ij} positive scalars.

$$e^B_{ij} = ||r_{ij}|| - D_{ij} \qquad 3.2.b$$

$$C^B(Q) = (1/p_B)\ \mathrm{SUM}_E[\ B_{ij}(e^B_{ij})^2\ ] \qquad 3.3.b$$

with p_B and B_{ij} positive scalars.

The summation in the expressions is over all edges (i,j) in the set of edges E of the underlying graph.

The edge weights can be used to scale the terms in the cost functions and affect the way in which they are edges are treated by the minimization procedure. They can be used to emphasize edges for which distance measurements are trusted better.

The following property is a straightforward consequence of these definitions.

Let P be the set of all Positions of the frame.

Property 1: $C(Q) \geq 0$. $C(Q) = 0$ <===> Q element of P.

In words: the cost functions are non-negative and they equal zero only at realizations compatible with the input frame.

From equations 3.2 and 3.3 it is obvious that C(Q) cannot be negative. The definitions of edge errors imply that C(Q)=0 only if $||r_{ij}||=D_{ij}$ for all (i,j) in E. Since by assumption T is rigid then this condition is equivalent to requiring that Q be an element of P. Sufficiency is shown similarly.

The value C(Q)=0 corresponds to a global minimum of the cost function. If the input frame is completely rigid there will be a unique global minimum (up to Isometric Transformations). If the input frame admits several Positions the global minima will correspond to each of these possible Positions. In practice the value of zero may not be attained and a suitably small value may have to be considered instead.

3.3 GRADIENT

The following symbols will be used in the derivation of expressions for the gradient of the cost functions.

x,y: Coordinates in the space of operation S^+.

r_{ix}: The x component of the position vector r_i. ix will also denote the component of the realization vector Q corresponding to the x component of position vector r_i.

Ni: The set of vertices adjacent to i in G(V,E).

n_i: The size of the set Ni.

$g_{ix}(Q)$: (Also denoted g_{ix}.) The partial derivative of the cost function with respect to ix.

We number the vertices in V from 1 to v and assume an arbitrary ordering of the coordinates in the space of operation S^+. We assume a pre-specified ordering (e.g. lexicographic) of the edges (i,j) in E. We also assume that in each set Ni, i in V, the elements (i,j) have some pre-specified ordering. The matrices and vectors defined for analysis are taken to be consistent with these orderings. We define the following vectors and matrices:

$g_i(Q) = [\ldots, g_{ix}, \ldots]^t$ m-vector of first order derivatives with respect to the components r_{ix}. It inhabits the space of operation.

$g(Q) = [.|.|g_i^t|.|.]^t$ mv-vector listing all g_i vectors. It is the gradient of the cost function in the function space.

$l_i(Q) = [\ldots, l_{ij}, \ldots]^t$ n_i-vector of components l_{ij}, j in Ni. l_{ij} will be assigned below.

$l(Q) = [.|.|l_i^t|.|.]^t$ vector listing all l_i vectors.

$$Z_i = \begin{pmatrix} & | & \\ \cdots & s_{ij} & \cdots \\ & | & \end{pmatrix}$$

m x n_i matrix. s_{ij} is an m-vector, j in Ni.

$$Z = \begin{pmatrix} \ddots & & \\ & \boxed{Z_i} & \\ & & \ddots \end{pmatrix}$$

Block diagonal matrix, Z_i: i in V.

Z has m x v rows (v= No of vertices) and the number of columns is:

$$\mathrm{SUM}_{i \text{ in } V}[n_i]$$

3.3.1 Derivation of gradient for $C^A(Q)$.

From equations 3.2.a and 3.3.a we have:

$$e^A_{ij} = ||r_{ij}||^2 - D_{ij}^{\ 2} \ , \ C^A(Q) = (1/p_A)\mathrm{SUM}_E[A_{ij}\ e^A_{ij}{}^2]$$

Then

$$g^A_{ix}(Q) = \ d/dix\{\ C^A(Q)\ \}$$

$$= \ (4/p_A)\ \mathrm{SUM}_{Ni}[\ A_{ij} e^A_{ij} r_{ijx}\ ] \qquad 3.4.a$$

$$g^A_i(Q) = \ (4/p_A)\ Z^A_i\ l^A_i \qquad 3.5.a$$

where the column vectors s_{ij} in the matrix Z^A_i are the edge vectors r_{ij} and the components in the vector l^A_i are given by $l^A_{ij} = A_{ij}\ e^A_{ij}$.

Equation 3.5.a shows that the part $g_i(Q)$ of the gradient corresponds in the space of operation S^+ to a weighted sum of the edge vectors incident on vertex i where the weight for vector r_{ij} is given by l^A_{ij}.

We can express the gradient of $C^A(Q)$ in matrix notation by

$$g^A(Q)= (2/p_A)\ Z^A\ l^A \qquad 3.6.a$$

Note that the elements in Z^A and l^A are not algebraically independent since they both depend on the position vectors. The notation however is good for expressing the conceptual relation among the relevant variables in the system (e.g. edge errors, position vectors, components of g(Q)).

3.3.2 Derivation of gradient for $C^B(Q)$.

From equations 3.2.b and 3.3.b:

$$g^B_{ix}(Q)= d/dix\{\ C^B(Q)\ \},\ e^B_{ij}= ||r_{ij}|| - D_{ij}$$

$$= (2/p_B)\ SUM_{Ni}[\ B_{ij}e^B_{ij}(r_{ijx}/||r_{ij}||)\ ] \qquad 3.4.b$$

$$g^B_i(Q)= (2/P_B)\ Z^B_i\ l^B_i \qquad 3.5.b$$

where the column vectors s_{ij} are the unit vectors u_{ij} pointing in the direction of the edge vectors and the vector l^B_i is given by

$l^B_{ij}= B_{ij}\ e^B_{ij}$. As before,

$$g^B(Q)= (2/p_B)\ Z^B\ l^B \qquad 3.6.b$$

As an example, for the choice $p_B= 2$, $B_{ij}= 1/D_{ij}$ we obtain $g^B{}_{i}(Q)= Z^B_i\ l^B_i$ and the components l^B_{ij} are simply the signed percentage errors $(||r_{ij}||-D_{ij})/D_{ij}$, (i,j) in E.

The expressions derived for the gradients suggest an analogy on the descent method. Consider first equation 3.6.b for the type B cost function. Let $p_B = 2$. The column vectors in Z^B_i are unit vectors and the weights 1^B_{ij} are given by B_{ij} e^B_{ij}.

The negative of the gradient vector points towards the direction of fastest decrease of $C^B(Q)$. If this negative of the gradient vector were used in equation 3.1 to minimize the value $C^B(Q)$ (steepest descent method) then the following analogy is possible:

The edges incident on vertex i in the space of operation can be considered to be springs of natural length D_{ij}. In any given realization the springs have lenghts $||r_{ij}||$.

If $||r_{ij}|| \neq D_{ij}$ the spring (i,j) will be under compression or extension by an amount e^B_{ij} (positive values of e^B_{ij} corresponding to extension). Hence the vector $-g^B_i(Q)$ can be considered the resultant force exerted on vertex i due to the deformation of the springs attached to it. The spring constants have value B_{ij} and the force exerted has magnitude B_{ij} e^B_{ij}.

The energy in the springs is

$$(1/2)\ B_{ij}\ e^{B\ 2}_{ij}$$

so the cost function $C^B(Q)$ (equations 3.3.b) is analogous to the total potential energy in the system. The algorithm is then gradually releasing this potential energy seeking the configuration of minimal energy.

Other choices of p_B lead to the same results since we need only modify the terms B_{ij} accordingly to obtain the exact analogy. For cost functions of type A the edge vectors will not correspond to springs in the usual sense. In this case we have special springs whose force is not proportional to their deformation.

The expressions derived for the components of the gradient in this section are used by the software produced for testing the iterative approach to Position Location.

3.4 STATIONARY POINTS

The following equation defines a stationary point of C(Q):

$$g(Q) = \text{0-vector} \qquad 3.7$$

equivalently (equations 3.5),

$$g_i(Q) = \text{0-vector} \qquad \text{for all } i \text{ in } V \qquad 3.8$$

Let Q' be a realization compatible with $T_n(G,D)$, then $g(Q')$=0-vector since $e_{ij}=0$ for all (i,j) in E. Q' is a solution of the PL problem and, since $C(Q')=0$, is a global minimum of C(Q).

Equation 3.8 will be satisfied at the point Q of the function space if the realization, also denoted Q, is such that the weighted vector sums expressed by equations 3.5 are equal to 0-vector. Pursuing the analogy proposed in the previous section: the resultant of forces acting on every vertex i must be zero.

lemma 3.1: If some edge error e_{ij} is non-zero a necessary condition for $g_i(Q)$= 0-vector, i in V, is that the edge vectors

r_{ij} be linearly dependent.

Proof: If the set of edge vectors r_{ij} for vertex i is linearly dependent so are the set of column vectors s_{ij} in the matrix $Z_i(Q)$ (for both cost functions), and viceversa, since s_{ij} equals u_{ij} or r_{ij} depending on the cost function (section 3.3).

If $g_i(Q)$= 0-vector and some edge error is non-zero then there is a linear combination of the vectors s_{ij} which gives the zero vector, namely that given by the coefficients l_{ij}. Hence the set of vectors s_{ij} is linearly dependent and so is the set of edge vectors r_{ij}.

Clearly this is not a sufficient condition. The scalars l_{ij} cannot be chosen freely, they depend directly on the edge vectors. Thus the edge vectors may well be linearly dependent yet $g_i(Q)$ may still not be equal to the 0-vector.

def 3.1: A "conflict point" of the cost function C(Q) is a stationary point of C(Q) which is not a global minimum.

This definition is for mere convenience, to facilitate references to such points. The choice of words comes from the analogy to springs. At a conflict point the configuration of springs would be such that their forces exactly cancel out at all vertices so that there are no resultant forces to move the vertices further and allow the springs to recover their natural lengths.

If Q is a conflict point then

$$g(Q)=0\text{-vector} \qquad \text{and} \qquad C(Q)\neq 0$$

Theorem 3.1: Cost functions of the type B are, in general, not globally convex. They may contain stationary points which are not global minima of the function.

Proof: The existence of conflict points in a cost function implies that the function is not globally convex. Below we show by construction of an example an instance (in one dimension) of a conflict point (a local minimum) for a simple frame, a triangle. Similar cases are imaginable in higher dimensions so we are led to the conclusion that there are instances of PL problems for which $C^B(Q)$ is not globally convex.

This argument also suggests that the same may be true of type A cost functions.

Example: Consider a 1-d input frame $T_1(G,D)$, $G(V,E)=K_3$.

$V= \{ 1,2,3 \}$

$E= \{(1,2),(2,3),(1,3)\}$

$D= \{ A,B,C \}$, with $A + B = C$.

Figure 3.1-a shows $T_1(K_3,D)$ and Figure 3.1-b shows a realization Q' which is a conflict point.

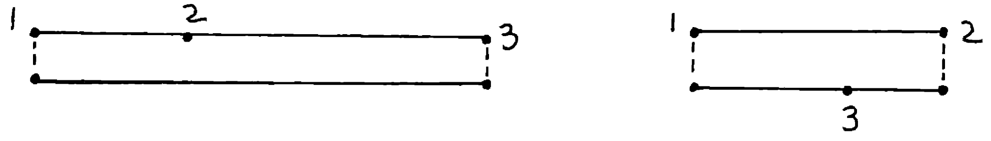
Let $||r_{12}||= a$, $||r_{23}||= b$ and $||r_{13}||= c$. Without loss of generality let the edge weights B_{ij} be equal to one for all edges. We obtain for $C^B(Q)$

$$g^B(Q) = \begin{pmatrix} -\frac{(a-A)}{A} - \frac{(c-C)}{C} \\ \\ \frac{(a-A)}{A} + \frac{(b-B)}{B} \\ \\ \frac{(c-C)}{C} - \frac{(b-B)}{B} \end{pmatrix} = \begin{pmatrix} 0 \\ \\ 0 \\ \\ 0 \end{pmatrix}$$

The first row of $g^B(Q)$ is dependent on the other two. Noting that in Q' (Figure 3.1-b) we have $c + b = a$ (Figure 3.1-b) we obtain

$$a= A(B+C)/C \qquad b= AB/C \qquad c= A$$

Figure 3.1-b is drawn for the case A=1,B=2,C=3, giving a=1+2/3, b=2/3 and c=1. In this case it is easy to appreciate that the conflict point is a local minimum for the function defined in one dimension. If the vertices were defined in two dimensions the same configuration corresponds to a saddle point (see section 3.5).



3.1-a 3.1-b

Figure 3.1: example of conflict for simple 1-d network

3.5 AUGMENTED SPACE OF OPERATION

In the example of the previous section the conflict point is a local minimum for the one dimensional frame represented, displacement of any of the points in the one dimensional space initially increases the value of the cost function. However, in this case it is apparent that if the frame were allowed to exist in a two dimensional space then a way to decrease the value of the cost function is to displace vertex 3 perpendicularly to the original space of operation. By augmenting the dimensions of the space of operation we turn the local minimum into an inflexion point from which there is a direction of descent for the value of the cost function. Moreover, once vertex 3 is not collinear with vertices 1 and 2 an algorithm using the gradient of the cost function to decrease the value of this cost function would be able to proceed iterating towards the solution of this problem. Using the physical analogy stated earlier, the conflict point becomes a point of unstable equilibrium for the system in two dimensions; allowing vertex 3 to be displaced in an orthogonal direction to the original space allows the spring corresponding to edge (1,3) (under extension) to return to its natural length and similarly for the springs under compression. This observation suggests that, at least in some instances of conflict points for PL problems, augmenting the dimensionality of the space of operation may open up a descent direction allowing convergence to proceed.

Two questions come up at this point. First, is it always possible to turn a non-optimal local minimum (i.e. a local minimum which is not a global minimum) into an inflexion (saddle) point of the function by augmenting the dimensions of the space of operation, and hence of the function space. If this were the case it would be of importance in showing convergence of the algorithm. We have no conclusive results in this area.

Second, for cases were an augmented space of operation is useful, what would be the maximum number (if any) of dimensions to which the space would have to be augmented in a worst case. Lemma 3.1 seems to suggest that the maximum number of dimensions required could be the maximum, over all vertices i, of the degree number d(i) of the vertices. Lemma 3.1 says that for a conflict to occurr the set of edge vectors incident on each vertex must be linearly dependent. Now, n linearly dependent vectors may span at most n-1 dimensions,hence if the vertex i, say, on which these edge vectors are incident is displaced in a direction orthogonal to the span of these vectors the component $g_i(Q)$ of the gradient will be non-zero. We have no guarantee though that this orthogonal direction constitutes a descent direction, this consideration belongs to the first question. Also we would have to consider all vertices involved in the conflict, instead of only a local situation, to be able to derive conclusions. However, at least from a local point of view, it would seem that a dimension equal to the maximum, over all vertices, of d(i) would be enough to solve local versions of a conflict if it is true that an orhtogonal direction provides a descent direction.

In the next chapter we will see that the gradient g(Q) can be expressed in a more compact form than that using the matrix Z defined in this chapter. The new form for equation 3.6 is

$$g^A(Q) = (4/p_A) \; J^t{}_M(Q) \; R \; A \; (M_G{}^2 - D^2)$$

$$g^B(Q) = (2/p_B) \; J^t{}_M(Q) \; B \; (M_G - D)$$

where $J_M(Q)$ is the Jacobian matrix of the measurement function with respect to the variables in the system (the coordinates of the vertices), matrices A and B are related to the edge weights for each cost function, and matrix R is related to the position of the vertices in the space of operation. It will also be the case that the Hessian matrix of the cost function is closely related to this same Jacobian matrix and that this Jacobian matrix is in turn closely related to rigidity. Using these expressions for the gradient and the expressions that will be derived for the Hessian matrix it may be possible to analyse the local behaviour of the cost function at conflict points by using Taylor's theorem to express a second order approximation to the cost function and investigating its local convexity properties. This may be a way to investigate the effect of augmenting the number of dimensions in the space of operation and help decide the first question mentioned above.

CHAPTER 4

In this chapter we introduce the Jacobian matrix of the measurement function with respect to the independent variables in the system (i.e. the position coordinates of the vertices in the space of operation). We explain its relevance to rigidity and revisit theorem 2.1 by Gluck. This topic is found in [Y 79]. We also explore the effect of a method for constraining the trivial displacements allowed on a frame. We refer to this method as "grounding" of the frame. It affects the null space of the Jacobian matrix mentioned and is of relevance to the strict convexity of the cost function near the solution points.

Next the Hessian matrix of the cost function is derived. The expressions for the entries in this matrix appear summarized in table 4.1. This expressions are used by the software described in section 4.9 that implements the minimization method for Position Location. We also derive matrix form equations for the Hessian matrix (table 4.2) and show its explicit dependence on the Jacobian matrix of the measurement function. By considering the structure of the Hessian matrix at points in the "function space" (the space S of realization vectors) that correspond to solutions of the PL problem, and exploiting its dependence on the Jacobian matrix, we show that the Hessian matrix for both types of cost functions is positive definite at these solution points. Hence we establish strict local convexity of the functions at the solutions. Finally we present and discuss the iteration method chosen and its current implementation for research.

We also include some of the results obtained through computer experiments with this method.

For a reference on linear algebra see for example [S]

4.1 JACOBIAN MATRIX OF THE MEASUREMENT FUNCTION

Consider the measurement function $M_G(Q)$ of a given realization Q. $M_G(Q)$ is a multi-valued function whose components are the lengths of the edge vectors in Q. We assume the ordering of these components to be consistent with that chosen for the set of edges of G.

Denote the length of edge vector r_{ij} in the realization Q by $M_{ij}(Q)$.

$$M_{ij}(Q) = ||r_i - r_j|| = ||r_{ij}|| \qquad 4.1$$

The argument Q emphasizes the dependence of these lengths on the realization. Let "e" be the number of edges in the graph G. The measurement function is then the e-vector given by:

$$M_G(Q) = \begin{pmatrix} . \\ . \\ . \\ M_{ij}(Q) \\ . \\ . \\ . \end{pmatrix} = \begin{pmatrix} . \\ . \\ . \\ ||r_{ij}|| \\ . \\ . \\ . \end{pmatrix} \qquad 4.2$$

The order of the subtractions in equations 4.2 is irrelevant.

$M_G(Q)$ is a function of all the independent variables in the system. More explicitly, it is a function of each coordinate variable of each vertex of the realization in the space of operation S^+. Note that if some of these coordinates were considered constant then the number of independent variables would decrease; this will be the case whith Grounding (section 4.5).

Let (k,l) be an edge of G. Let "grad $M_{kl}(Q)$" be the vector of first order partial derivatives of $M_{kl}(Q)$ with respect to the independent variables in the realization. Let F be the number of independent variables. The Jacobian matrix of $M_G(Q)$ with respect to these variables, denoted by $J_M(Q)$, is an e by F matrix given by:

$$J_M(Q) = \begin{bmatrix} \\ \\ \text{-------}\ \text{grad}^t\ M_{kl}(Q)\ \text{-------} \\ \\ \\ \end{bmatrix} \qquad 4.3$$

If all coordinates are unconstrained $J_M(Q)$ is a matrix of dimensions e by mv where "m" is the dimension of the space of operation S^+ and "v" is the number of vertices in G. The entry of $J_M(Q)$ corresponding to row "(k,l)" and column "ix" is given by

$$d/dix\{\ ||r_k - r_l||\ \} = \begin{cases} r_{klx}/||r_k - r_l|| & \text{for } i=k \\ -\ r_{klx}/||r_k - r_l|| & \text{for } i=l \\ 0 & \text{for } i\neq k,\ i\neq l \end{cases}$$

(d/dix denotes partial derivative w.r.t. ix).

Denoting again by u_{ij} the unit vector in direction r_{ij}:

$$d/dix\{ ||r_k - r_l|| \} = \begin{cases} u_{klx} & \text{for } i=k \\ -\, u_{klx} & \text{for } i=l \\ 0 & \text{for } i\neq k,\ i\neq l \end{cases} \tag{4.4}$$

We can illustrate the structure of $J_M(Q)$ with an example. Consider the frame of Figure 4.1 in two dimensional space. Its Jacobian matrix appears below.

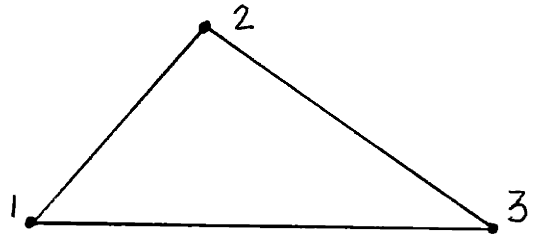


Figure 4.1: two-dimensional frame for Jacobian matrix example

Jacobian matrix $J_M(Q)$:

$$\begin{array}{c|cccccc|}
 & 1x & 1y & 2x & 2y & 3x & 3y \\
\hline
(1,2) & u_{12x} & u_{12y} & -u_{12x} & -u_{12y} & & \\
(2,3) & & & u_{23x} & u_{23y} & -u_{23x} & -u_{23y} \\
(3,1) & -u_{31x} & -u_{31y} & & & u_{31x} & u_{31y} \\
\hline
\end{array}$$

We can write this in the block matrix form below, where entries stand for row vectors.

$$\begin{array}{c|ccc|}
 & 1 & 2 & 3 \\
\hline
(1,2) & u_{12}^{t} & -u_{12}^{t} & \\
(2,3) & & u_{23}^{t} & -u_{23}^{t} \\
(3,1) & -u_{31}^{t} & & u_{31}^{t} \\
\hline
\end{array}$$

A simple way to write down the $J_M(Q)$ is to construct it as an incidence matrix where the entries are row unit vectors. The entry corresponding to row "(k,l)" and block column i contains the unit edge vector for (k,l) pointing towards i if the edge (k,l) is incident on vertex i. The entry would be a zero-vector (row vector) if (k,l) is not incident on i.

Any row has exactly two unit edge vector entries and the rest of the vector entries are zero entries. Any block column contains all the unit edge vectors incident on the vertex corresponding to this column.

4.2 RANGE AND NULL SPACE OF JACOBIAN MATRIX

The function $M_G(Q)$ (equation 4.2) is completely continuously differentiable w.r.t. the independent variables. Let the vector "q" denote a small increase in these variables, then, from Taylor's theorem, we have to a first order approximation:

$$M_G(Q+q) = M_G(Q) + J_M(Q)q \qquad 4.5$$

For small enough values of q the second term in the expression, the vector $J_M(Q)q$, is approximately equal to the increase in the length of the edge vectors in the realization induced by displacing the position vectors of the vertices by an amount specified by the vector q, (first order approximation). In particular, the null space of $J_M(Q)$ corresponds to small alterations to the realization that do not alter the length of the edge vectors, not by virtue of their being small but by virtue of the displacement directions chosen for the vertices in these alterations.

The range of $J_M(Q)$ corresponds to the possible increases in $M_G(Q)$ that can be achieved by a suitably small alteration of the realization. To illustrate this consider a frame with realization Q whose underlying graph is a triangle. $M_G(Q)$ is a vector whose component values are equal to the length of the sides of the triangle. If one chooses some arbitrary small displacement for the vertices in the realization, q say, then the range of $J_M(Q)$ denotes, for small values of q, the way in which the sides of the frame will be altered. For example, an alteration on the edge vectors of Q above, however small in magnitude, that results in sides which do not satisfy the triangle inequality will be outside the range of $J_M(Q)$ since there is no way one could displace the vertices of the frame above and achieve such an alteration on its sides.

4.3 TEST FOR INFINITESIMAL RIGIDITY

Consider a velocity assignment w(t) on the vertices of a realization, where the argument t is a continuous scalar variable, such that, for small increments dt and dq of the argument t and of the realization vector Q respectively, we have

$$w(t)\ dt = dq \qquad\qquad t \text{ a scalar}$$

For infinitesimally small values of t the assignment tends to $w_0 = w(0)$ and in the limit

$$J_M(Q)\ dq = J_M(Q)\ w_0\ dt$$

Since dt is only a scalar the null space of $J_M(Q)$ can also be taken to correspond to velocity assignments, w(t) say, on the vertices of the realization such that in the limit, as t -> 0 and w(t) -> w_0, the edge lengths in the realization are not altered.

Now, for a given graph G(V,E) a realization Q(V) defines a frame $T_m(G,D(Q))$, where $D(Q) = M_G(Q)$, so for this frame the null space of $J_M(Q)$ corresponds to the directions of motions that can be assigned to the vertices of the frame which preserve the length of edge vectors. Any velocity assignment equivalent to a trivial displacement of the frame lies in the null space of $J_M(Q)$.

This leads to a test of infinitesimal rigidity in the following way. Suppose we are given a realization Q, contained in two dimensions, of a frame $T_2(G,D)$. In two dimensions there exist exactly three independent trivial displacements for this frame: two are the orthogonal translations (e.g. vertical and horizontal) and the last is the motion of rotation of all points around the origen. These three displacements must lie in the null space of $J_M(Q)$. With no constraints on the variables the matrix $J_M(Q)$ has dimensions e by 2v so the rank of this matrix is <u>at most</u> 2v-3. Furthermore, if the rank is less than 2v-3 then there must be some other velocity assignment on the vertices which is not a trivial displacement and yet preserves edge lengths. Hence (by definition 2.9) the frame is not infinitesimally rigid.

(Compare these results with theorems 2.3 and 2.4 by Laman).

A similar analysis applies to m dimensions ([AR],[Y 79]). In m dimensions let the number of independent trivial displacements of a given frame be "f". If the frame is itself m-dimensional f equals the number of degrees of freedom available in an m-dimensional space, which is [AR]

$$f = m(m+1)/2 \qquad 4.6$$

otherwise f may assume some smaller value.

In either case if the rank of the Jacobian matrix is less than

$$F = mv - f \qquad 4.7$$

the null space comprises some non trivial displacements. Lemma 4.1 follows from this discussion.

<u>lemma 4.1</u>: The frame T(G,D) of realization Q is infinitesimally rigid in m dimensions if and only if

$$\text{rank of } J_M(Q) = mv - f \qquad 4.8$$

where f is the number of independent trivial displacements of the frame in m dimensions.

The rank of a matrix can be found by polynomial time algorithms like Gaussian elimination. This is convenient in computational terms.

Infinitesimal rigidity implies rigidity (see section 2.4). However a rigid frame may not be infinitesimally rigid, consider the example of figure 4.2. The velocity assignment corresponds to moving only vertex 2 in the direction shown by the dotted arrow. This is not a trivial displacement for the frame yet, in the limit as t -> 0, the displacement preserves edge lengths because the assignment to vertex 2 is perpendicular to both edges incident on 2.

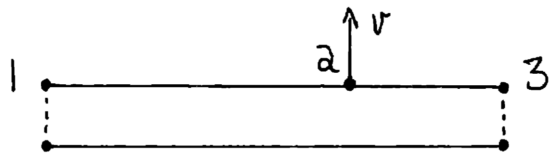


Figure 4.2: rigid frame which is not infinitesimally rigid in two or more dimensions.

Because of its importance we state these facts as a lemma.

lemma 4.2: An infinitesimally rigid frame is rigid. A rigid frame is not necessarily infinitesimally rigid.

At this point we can recall theorem 2.1 by Gluck about infinitesimal rigidity and outline its proof. We state again the theorem here.

Theorem 2.1: Let G(V,E) be a graph of v vertices. Let the position vectors of the vertices in a realization of G(V,E) be R^n, and identify the space of all possible realization vectors with R^{nv}.

Then if there is a frame $T_n(G,D_0)$ which is infinitesimally rigid, the set of frames $T(G,D)$ (with $D=D(G)$ arbitrary), of n dimensions or less, which are infinitesimally rigid corresponds to an open and dense set of realization vectors in R^{nv}.

Outline of Proof ([Y 79],[Gl]):

Let $D = M_G(Q)$ for $Q=Q^*$. Let $F = nv - n(n+1)/2$. From lemma 4.1, $T_n(G,D)$ is infinitesimally rigid if and only if the rank of $J_M(Q^*) = F$. This implies that some F by F subdeterminant of $J_M(Q^*)$ is not equal to zero.

Suppose we sum the square of all F by F subdeterminants of $J_M(Q)$. Then the frame is infinitesimally rigid if and only if this polynomial is not zero at Q^*. But if there is some realization such that this polynomial is not zero then the polynomial cannot be identically zero. The realizations which correspond to infinitesimally rigid frames are those for which this polynomial is non-zero, and this corresponds to an open and dense set in the function space R^{nv}. Q.E.D.

In this sense there can be degenerate cases of non-infinitesimally rigid frames; those for which the polynomial above becomes zero.

4.4 TEST FOR RIGIDITY

A test for infinitesimal rigidity is not exactly equivalent to a test for rigidity (c.f. lemma 4.2). Bearing theorem 2.1 in mind however a polynomial time probabilistic decision algorithm for rigidity can be created. This algorithm is presented and discussed in [Y 79] and we describe it here informally.

We are given a frame $T_n(G,D)$ but no realization of it. Suppose we now want to answer QUESTION 1 of the PL problem (section 2.3) i.e. we want to test for rigidity before trying to find a Position of T. A test for infinitesimal rigidity cannot be directly applied since that would require knowing a compatible realization of T (i.e. a Position). Instead we generate an arbitrary realization of the underlying graph G. This can be done for example by drawing a random realization point Q from a uniform distribution over a set of points R in the space R^{nv}. We test for infinitesimal rigidity of the realization drawn. If the test result is affirmative the algorithm declares $T_n(G,D)$ to be rigid. If the test result is negative the algorithm declares it not rigid. Using theorem 2.1 it can be shown [Y 79] that such an algorithm decides correctly with probability as close to 1 as desired by increasing the number of points in the set R. With the size of R chosen to be a polynomial in the number of vertices the algorithm will stop in polynomial time.

4.5 GROUNDING

Even for a rigid frame, if all coordinates of all vertices are unconstrained in the function space S, there will be a whole class (a continuum) of equivalent realizations compatible with the frame.

They arise from the infinitely many trivial displacements that the frame can have. In terms of solving the Position Location problem it would be more suitable if, for a rigid frame, we had, instead of a continuum of solutions in S, a point-like solution. This would correspond to having fixed the Position of the frame in the space of operations so that trivial displacements of the frame are supressed.

Let f be the number of independent continuous trivial displacements for a rigid frame $T_n(G,D)$ in the space of operations. If we can constrain the locus of this frame so as to supress some degrees of freedom we will reduce accordingly the number of independent trivial displacements possible on the frame.

We call "Grounding" one such method by which the f possible independent continuous trivial displacements can be supressed when a Maximally Spanning set of n+1 vertices (definition 2.11) is present in the frame. As will be explained below, Grounding is an operation on n+1 vertices and will achieve the desired result if these vertices form a Maximally Spanning set. To ensure that Grounding will work, then, we should test if the set of vertices selected is indeed Maximally Spanning. An algorithm would have to scan the set of vertices of the frame to select a Maximally Spanning set, but of course we do not know if a set will be Maximally Spanning before we have obtained a Position of the frame. This represents a potential problem to the operation of an algorithm using Grounding. In the software described at the end of this Chapter we select the vertices for Grounding without any tests on the assumption that they will be Maximally Spanning. We describe Grounding below and then show how it achieves the result desired.

Consider the graph G(V,E) and the frame $T_n(G,D)$. Let v be the number of vertices in V.

Grounding (for n dimensions):

1- Choose a set V_{n+1}, subset of V, of n+1 vertices.

Without loss of generality, assume that the vertices in V_{n+1} are labeled from 0 to n+1.

2- Constrain vertex 0 to lie at a fixed position in R^n.

3- Constrain the ith vertex of V_{n+1}, i=1,...,n+1 , to lie on an affine

space S_i, of i dimensions, containing the affine space S_{i-1}. S_0 is defined to be the position of vertex 0.

For the frame to be n dimensional it must have at least n+1 vertices (c.f. section 2.5.1) which is the dimensionality of V_{n+1}, hence step 1 above is always possible.

For convenience we will refer to the set V_{n+1} used in the process of Grounding as "the Root" of the frame. Again, without loss of generality, we assume for simplicity that the first i coordinate axes of the vector space defined by the Root correspond to those of the i-dimensional space S_i in the Grounding process, where i<k, and that the origin of these axes is vertex 0 of the Root. Thus, Grounding is achieved by constraining all coordinates of vertex 0 of the Root to be zero, all but the first coordinate of vertex 1 of the Root to be zero, and so up to vertex k of the Root. This means that if $T_n(G,D)$ is grounded in the space of operations several vector components of the vertices of T must be held constant (equal to zero)

under any displacement of $T_n(G,D)$. This reduces the number of independent variables in the system.

There will be exactly f less variables in the system, where f equals the number of possible independent continuous trivial displacements. Let F be the number of independent variables in the system after Grounding it. Then if the space of operation has m dimensions F is given by:

$$F = mv - f$$

with f given by equation 4.6 if n=m.

The space of operation S^+ would still be an m-dimensional space, where the loci of some vertices have been constrained, but the function space S is now reduced to F dimensions, from mv dimensions that it would have if no Grounding were implemented, since S is a space defined on the independent variables of the system. Similarly, the cost function, after Grounding, is now a map $C(Q)\colon R^F \to R$.

It will be seen below, in the proof of Theorem 4.1, that for a grounded , infinitesimally rigid frame whose Root is Maximally Spanning the possible solutions in the space S are point like as desired. This can be related to the null space of the Jacobian matrix of the Measurement function at the solutions and becomes relevant to the convexity properties of the cost function.

<u>Theorem 4.1</u>: Let the frame $T_n(G,D)$ be infinitesimally rigid and grounded in n dimensions. Let the Root of $T_n(G,D)$ be Maximally Spanning, and let $J_M(Q^*)$ be the Jacobian matrix of the

Measurement function $M_G(Q^*)$, where Q^* is a (grounded) realization compatible with $T_n(G,D)$. Then the null space of $J_M(Q^*)$ is empty.

In the proof of this Theorem we will require lemma 4.3 below:

Let V_{k+1} be a Maximally Spanning set of k+1 vertices in a k-dimensional vector space. Let V_k be a subset of k vertices of V_{k+1} whose position vectors P_i, $i=1,\ldots,k$, are known. (V_k is necessarily Maximally Spanning.) Let R be the unknown position vector of the vertex of V_{k+1} not in V_k. Let distances $D_i=||R-P_i||$, $i=1,\ldots,k$, be known. Let Z be the affine space spanned by the set of vectors $\{P_{ij}$: i,j in $V_k\}$. (Z has k-1 dimensions.)

Then:

<u>lemma 4.3</u>: The position vector R is determined up to a reflection by the given vectors P_i and the given distances D_i. The reflection is through the affine space Z.

Note: This lemma is simply a generalization of the solution to the system of equations 1.1 seen in the introduction.

Proof: Let the components of R be $R_1,\ldots,R_k$ and the components of P_i be $P_{i,1},\ldots P_{i,k}$. R satisfies the set of k equations:

$$||r - p_i||^2 = D_i^2 \qquad i=1,\ldots,k$$

Select any equation i and subtract from it the remaining equations j, j≠i. We obtain the set of k-1 equations:

$$(P_{j,1}-P_{i,1})\ R_1 + \ldots\ldots + (P_{j,k}-P_{i,k})\ R_k = K(P_j,P_i)$$

$$\text{for } j= 1,\ldots,k,\ j\neq i$$

where $K(P_j,P_i)$ is a constant found from the distances and position vectors known.

We write this system of equations in matrix notation as

$$P\ R = K$$

where P is a k-1 by k matrix, R is the unknown position vector, and K is a vector of k-1 components (known).

We notice that the k-1 row vectors of P form a basis for Z, since they consist of the set of difference vectors P_{ji} for all j≠i. Hence, from the dimensions of P, the null space of P must be 1-dimensional. Thus the set of solutions for the matrix equations can be written as $R^{\&} + tX$, where $R^{\&}$ is any solution to the matrix equation, X is any vector in the null space of P, and t is any scalar. All solutions to the matrix equation above must lie on the line defined by $R^{\&} + tX$.

The solutions to our original set of k equations will lie on this line and also on the hyperspheres defined by the distances given.

Now, the intersection of a line and a hypersphere is either empty, a single point, or two disjoint points. Since our system must have at least one solution the intersection in this case cannot be empty. Since V_{k+1} is Maximally Spanning R must lie outside Z. Thus, if R_a is a solution of the original system of k equations then the reflection of this solution through Z also satisfies the same distance constraints and hence must also be a solution. Hence, the vector R is determined, up to a reflection as explained, by the parameters given, which proves lemma 4.3.

Note: if only V_k were required to be Maximally Spanning then it would also be possible to have a single point as the intersection of the line and the hyperspheres above. This situation would mean that R in fact lies in the space defined by V_k.

Proof of Theorem 4.1:

If T_n is infinitesimally rigid it is also rigid (c.f. section 2.4). Hence any continuous displacement of $T_n(G,D)$ must satisfy

$$M_{Kv}\{Q(t)\} = M_{Kv}\{Q(0)\} = M_{Kv}\{Q^*\}$$

i.e. all distances between any two vertices of the frame must remain invariant under a continuous displacement.

If $T_n(G,D)$ is grounded then there is a vertex in the Root which is fixed at the origin. Another vertex is constrained to lie on the X_1 axis of the space of operation S^+. By lemma 4.3 and since distances between vertices in the Root are constant, the position of this second vertex is completely determined up to a reflection and so invariant under a continuous displacement. By induction (using lemma 4.3 and the fact that distances between vertices must be constant) on the rest of the vertices of the Root we can show that the position of all vertices of the Root are fixed (up to reflections) and so must remain invariant under continuous displacements. As a simple extension of lemma 4.3, the positions of the remaining vertices of the frame are uniquely determined by their distances to the vertices in the Root (once the reflections in the Root have been chosen) and so the positionsmust also remain invariant under any continuous displacement. Thus no continuous displacement of $T_n(G,D)$ (trivial or otherwise) is possible once it is grounded, on the assumption that it is infinitesimally rigid and that the Root chosen is Maximally Spanning.

The Jacobian matrix $J_M(Q^*)$ of the cost function for the grounded case has a null space corresponding to trivial displacements of the frame since the frame is infinitesimally rigid by assumption. But, as shown above, no trivial displacements are now possible. Hence the null space of the $J_M(Q^*)$ must be empty. Q.E.D.

Notice that all possible solutions must be point-like in the space of realization vectors S as was desired.

def 4.1: A frame is said to be "Properly Grounded" if it is grounded and the Root is Maximally Spanning.

4.6 HESSIAN MATRIX OF THE COST FUNCTIONS

The Hessian matrix of C(Q) is the matrix of second order partial derivatives with respect to the free variables in C(Q). It is a symmetric matrix of dimensions mv by mv when no grounding is used (for the grounded case the dimensions are F by F). It is a function of Q and we denote it by H(Q). For simplicity of notation define

$$H_{ixjy} = H_{ixjy}(Q) \equiv d/djy\{\ d/dix\{C(Q)\}\ \} = H_{jyix}$$

$$= d/djy\{\ g_{ix}(Q)\ \}$$

We consider separatedly five different situations when obtaining expressions for the elements H_{ixjy} of the matrix H(Q); when i=j we have the case x=y and the case x≠y; when j is an element of Ni we also have these two possibilities, and finally there is the situation when j≠i and j is not in Ni. Table 4.1 contains a summary of the expressions derived. Derivation of the expressions obtained appears in the appendix of this thesis. These expressions are used in the software described in section 4.9.

The Hessian matrices obtained have an interesting structure which will be developed after presenting table 4.1. These results are summarized in table 4.2. Notation is the same as for chapter 3. u_{ij} is the unit vector in the direction of r_{ij}.

TABLE 4.1 summary of equations for the elements of H(Q)

TYPE A COST FUNCTION

<u>case 1</u>: $j=i$, $x=y$

$$H^A_{ixjy} = (4/p_A)\ \mathrm{SUM}_{k \text{ in } Ni}[\ A_{ik}(\ e_{ik} + 2(r_{ikx})^2\ )\ ] \qquad 4.9.a$$

<u>case 2</u>: $j=i$, $x \neq y$

$$H^A_{ixjy} = (8/p_A)\ \mathrm{SUM}_{k \text{ in } Ni}[\ A_{ik}(\ r_{ikx} r_{iky}\ )\ ] \qquad 4.10.a$$

<u>case 3</u>: j in Ni, $x=y$

$$H^A_{ixjy} = -(4/p_A)\ [\ A_{ij}(\ e_{ij} + 2\ r_{ijx}{}^2\ )\ ] \qquad 4.11.a$$

<u>case 4</u>: j in Ni, $x \neq y$

$$H^A_{ixjy} = -(8/p_A)\ A_{ij}\ r_{ijx} r_{ijy} \qquad 4.12.a$$

<u>case 5</u>: $j \neq i$, j not in Ni

$$H^A_{ixjy} = 0 \qquad 4.13.a$$

TABLE 4.1 (continued)

TYPE B COST FUNCTION

case 1: j=i , x=y

$$H^{B}_{ixjy} = (2/p_B)\mathrm{SUM}_{Ni}[B_{ik}\{(u_{ikx})^{2}+(1-u_{ikx}{}^{2})e_{ik}/||r_{ik}||\}] \qquad 4.9.b$$

case 2: j=i , x≠y

$$H^{B}_{ixjy} = (2/p_B)\ \mathrm{SUM}_{k\ in\ Ni}[\ B_{ik}\ u_{ikx}u_{iky}\ ] \qquad 4.10.b$$

case 3: j in Ni, x=y

$$H^{B}_{ixjy} = -(2/p_B)[B_{ij}\{u_{ijx}{}^{2}+(1-u_{ijx}{}^{2})e_{ij}/||r_{ij}||\}] \qquad 4.11.b$$

case 4: j in Ni, x≠y

$$H^{B}_{ixjy} = -(2/p_B)\ [\ B_{ij}\ u_{ijx}u_{ijy}\ ] \qquad 4.12.b$$

case 5: j≠i , j not in Ni

$$H^{B}_{ixjy} = 0 \qquad 4.13.b$$

In the appendix it is shown that the Hessian matrix of the cost functions can be written in the form:

$$H^A(Q)= (4/p_A)\ (2\ P^A + E^A) \qquad 4.14.a$$

$$H^B(Q)= (2/p_B)\ (P^B + E^B) \qquad 4.14.b$$

This expresses the matrix H(Q) as the sum of two matrices, one which depends solely on the position vectors of the vertices (matrix P) and the other which also depends on the edge errors e_{ij} (matrix E). In the appendix it is shown that the matrix E contains the edge errors e_{ij} as factors of all its elements and hence becomes zero when the edge errors are zero (i.e. when the position of the vertices form a realization compatible with the frame).

Let us now obtain expressions for the structure of the Hessian matrices of both cost function types.

Consider a scalar function f of several variables. Let the n-vector X denote these variables and let $f(X)=a^tb$ where a and b are m-vectors whose components are functions of X. Let J(F) denote the Jacobian w.r.t. X of a function F(X). Then it is straightforward to show that

$$J(f(X)) = J(a^tb) = a^tJ(b) + b^tJ(a) \qquad 4.15$$

Also, if A is an "m by k" matrix of column vectors a_i whose components depend on X then it can also be shown that

$$J(A^tb)= A^tJ(b) + \begin{bmatrix} b^t & & & & \\ & b^t & & & \\ & & \cdot\cdot & & \\ & & & \cdot\cdot & \\ & & & & b^t \end{bmatrix} \begin{bmatrix} J(a_1) \\ J(a_2) \\ \cdot \\ \cdot \\ J(a_m) \end{bmatrix} \qquad 4.16$$

The matrix on the left of the second term is a block diagonal matrix of dimensions k by m^2 and block diagonal entries b^t. The matrix on the right of the second term is a block diagonal matrix of dimensions m^2 by n and block entries $J(a_i)$ (dimensions m by n). We can write

$$J(A^tb) \quad = \quad A^tJ(b) \quad + \quad [b^t]^* \quad \{A\}^*$$

4.17

where $[b^t]^*$ and $\{A\}^*$ stand for the matrices on the left and right discussed immediately above.

For the case $f(X)= a^ta$:

$$g \equiv \text{grad } f(X) = J^t(a^ta)$$
$$= 2\ J^t(a)\ a$$

$$H \equiv \text{hessian of } f(X) = J(g)$$
$$= 2\ J^t(a)\ J(a) + 2\ [a^t]^*\ \{J(a)\}^*$$

We can now apply this results to our cost functions. For simplicity consider first $C^B(Q)$. We can write (c.f. equation 3.3.b)

$$C^B(Q)= (1/p_B)(M_G-D)^t\ B\ (M_G-D)$$

where $M_G=M_G(Q)=$ the measurement function, $D=D(G)=$ distance set vector, and B is a diagonal matrix of entries B_{ij}.

Using the previous equalities:

$$g^B(Q)= (2/p_B)\ J^t{}_M(Q)\ B\ (M_G-D) \qquad 4.18.b$$

$$H^B(Q)= (2/p_B)\ J^t{}_M(Q)\ B\ J_M(Q) + (2/p_B)\ [(M_G-D)^t\ B]^*\ \{J_M(Q)\}^* \qquad 4.19.b$$

where $J_M(Q)$ is the Jacobian matrix of the measurement function defined in section 4.1.

For $C^A(Q)$ we note, from equation 3.3.a, that

$$C^A(Q) = (1/p_A)\ (M_G{}^2-D^2)^t\ A\ (M_G{}^2-D^2)$$

where $M_G{}^2$, D^2 are vectors whose components are the square of the corresponding components in the vectors M_G, D , and A is a diagonal matrix of entries A_{ij}. Also note that, denoting by $J(M_G{}^2)$ the Jacobian matrix of $M_G{}^2$ w.r.t. the independent varables in the given realization, we have

$$J(M_G{}^2) = (2/p_A)\ R\ J(M_G)$$

where R is a diagonal matrix of entries $||r_{ij}||$. Hence,

$$g^A(Q)= (4/p_A)\ J^t{}_M(Q)\ R\ A\ (M_G{}^2-D^2) \qquad 4.18.a$$

$$H^A(Q)= (8/p_A)\ J^t{}_M(Q)\ R\ A\ R\ J_M(Q) + (4/p_A)\ [(M_G{}^2-D^2)^t AR]^*\ \{J_M(Q)\}^* \qquad 4.19.a$$

These results are summarized in table 4.2 below.

Table 4.2 summary of expressions for gradient and Hessian matrices (the operators $[\ ]^*$ and $\{\ \}^*$ have been defined above) (matrices R, A and B have also been defined above)

$$J(f(X)) = J(a^t b) = a^t J(b) + b^t J(a) \qquad 4.15$$

$$J(A^t b) = A^t J(b) + [b^t]^* \{A\}^* \qquad 4.17$$

$$g^A(Q) = (4/p_A)\ J^t{}_M(Q)\ R\ A\ (M_G{}^2 - D^2) \qquad 4.18.a$$

$$H^A(Q) = (8/p_A)\ J^t{}_M(Q)\ R\ A\ R\ J_M(Q) + (4/p_A)\ [(M_G{}^2 - D^2)^t\ A\ R]^*\ \{J_M(Q)\}^* \qquad 4.19.a$$

$$g^B(Q) = (2/p_B)\ J^t{}_M(Q)\ B\ (M_G - D) \qquad 4.18.b$$

$$H^B(Q) = (2/p_B)\ J^t{}_M(Q)\ B\ J_M(Q) + (2/p_B)\ [(M_G - D)^t\ B]^*\ \{J_M(Q)\}^* \qquad 4.19.b$$

By comparing these results to equations 4.14 we can identify the matrices P and E (equations 4.14) and express them in terms of the Jacobian matrix of the measurement function as

$$P^A = J^t\ T^A\ J$$

$$E^A = [(M_G{}^2 - D^2)^t\ A\ R]^*\ \{J_M(Q)\}^*$$

$$P^B = J^t\ T^B\ J$$

$$E^B = [(M_G - D)^t\ B]^*\ \{J_M(Q)\}^*$$

where, for convenience, T^A and T^B are diagonal matrices of dimensions e by e (e= number of edges) and diagonal components "(i,j)" given by

$$T^A{}_{ij} = A_{ij} ||r_{ij}||^2 \qquad 4.20.a$$

$$T^B{}_{ij} = B_{ij} \qquad 4.20.b$$

Note that since A_{ij}, B_{ij} and D_{ij} are always greater than zero then T_{ij} is always greater than zero.

We can now write the Hessian matrices in terms of the Jacobian matrix defined for the measurement function. We have

$$H^A(Q) = (4/p_A)\ (2\ J^t\ T^A\ J + E^A) \qquad 4.21.a$$

$$H^B(Q) = (2/p_B)\ (J^t\ T^B\ J + E^B) \qquad 4.21.b$$

We recall that matrices E in these expressions become zero matrices when Q is a realization compatible with the given frame since all edge errors become zero as discussed above.

4.7 CONVEXITY OF C(Q)

The effect of grounding the frame $T_n(G,D)$ is to supress degrees of freedom which are not interesting to our study. We show below that if the frame T is infinitesimally rigid then grounding will have the effect of making C(Q) locally strictly convex at the solution points Q (C(Q)=0).

Theorem 4.1: Let $T_n(G,D)$ be infinitesimally rigid and Properly Grounded (definition 4.1) in an n-dimensional space of operation and let Q' be a point in the (reduced) function space S such that

$$C(Q') = 0$$

Then, H(Q),the Hessian matrix of the cost function C(Q), Q a point in S, is positive definite at Q'.

Proof:

Equations 4.21 contain expressions for the Hessian matrix of the cost functions.

If $C(Q')=0$ then all edge errors at Q' equal zero. Hence the matrix E equals the 0-matrix (c.f. equations 4.14 and 4.15) and we have, from 4.21:

$$H^A(Q')= (p^A)\ (J^t\ T^A\ J) \qquad 4.22.a$$

$$H^B(Q')= (p^B)\ (J^t\ T^B\ J) \qquad 4.22.b$$

where $p^A= (8/P_A)$ and $p^B= (2/p_B)$.

Generalizing for both cost functions:

$$H(Q')= p\ (J^t\ T\ J) \qquad 4.22$$

From section 4.5, since $T_n(G,D)$ is assumed infinitesimally rigid and Properly Grounded and Q' is compatible with it, it follows that the null space of $J_M(Q')$ is empty.

From equation 4.22,

$$Y^t\ H(Q')\ Y\ =\ p\ (Y^t\ J^t\ T^{1/2}\ T^{1/2}\ J\ Y)$$

$$=p\ (\ T^{1/2}\ J\ Y\ )^2 \qquad 4.23$$

Since $T^{1/2}$ exists and is invertible (c.f. equations 4.20), and the null space of J is empty then

$$Y^t H(Q')Y\ \ \leq 0 \qquad 4.24$$

with equality if and only if $Y=$ 0-vector. Hence $H(Q')$ is a positive definite matrix. Q.E.D.

Note that if the input frame is rigid the cost functions can be locally strictly convex at the solution points even if the frame is not infinitesimally rigid. This is formally stated on the theorem below.

<u>Theorem 4.2</u>: Let $T_n(G,D)$ be rigid and Properly Grounded in an n-dimensional space of operation and let Q' be a point in the (reduced) function space S such that

$$C(Q')=0$$

Then:

Q' is an element of the set of Positions of the frame T.

Q' is a strict global minimum of the cost function $C(Q)$.

Proof:

The statements that Q' is an element of the set of Positions of the frame and a global minimum of the cost functions is simply property 1 of section 3.2.

To show that it is a strict global minimum note that if the frame is Properly Grounded no continuous trivial displacements are possible, hence no other realizations in the Position corresponding to Q' can have realization vectors contiguous to Q'. Since the frame is rigid the set of Positions of the frame is countable and thus no other realization compatible with T can have realization vectors contiguous to Q' either. Hence the neighbourhood of Q' contains only points for which $C(Q)>0$ and Q' is a strict (global) minimum.

Q.E.D.

We can note that since equation 4.23 is true even without grounding then $C(Q)$ is also locally convex at Q' when no grounding takes place, though not strictly convex.

It might be necessary to take provisions so that near enough to the solutions the algorithm takes action to force Q_{END} (the answer arrived at by the algorithm) to lie entirely in the subspace S_0 of S, i.e to lie in the space of solutions. For example, near enough to the solution the algorithm could decide to assign to the realization Q_{END} a value obtained by projecting the m-dimensional position vectors in the space of operation S^+ onto the space of solutions

S^{+}_{0} and obtaining the equivalent realization vector.

Another way could be to modify the cost function C(Q) once the algorithm is near enough to the solutions so that an error term is added to the cost function for components of the position vectors in S^{+} orthogonal to S^{+}_{0}.

These provisions constitute the remaining set of special provisions referred to in section 3.1 to make Q_{END} lie in the space of solutions.

4.8 MINIMIZATION METHOD

The minimization method chosen is explained in detail in the technical report "Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach" by David M. Gay [G 80] and the software is available from the Numerical Analysis Library (nalib), Information Processing Services, MIT. The software is also available from the Association for Computer Machinery (ACM) and explained in [G 83]. In this section we only review in general terms the principles of the method. For a reference on numerical methods for unconstrained optimization see [DS].

The cost functions we have chosen are maps C(Q): S->R, where S is the function space and R is the set of real numbers. The dimensions of S depend on whether Grounding has been implemented or not (c.f. section 4.5). The solutions to the PL problem correspond to global minima of the cost functions. The cost functions are analytic and twice continuously differentiable. We have a nonlinear unconstrained optimization problem:

$$\underset{Q \text{ in } S}{\text{minimize}} \quad C(Q)$$

In section 4.7 C(Q) was shown to be locally strictly convex at global minima when Properly Grounded. In section 3.4 we argue that the cost functions are not globally convex because of the existence of stationary points other than global minima (conflict points). Finally, we have available analytic expressions for the gradient and Hessian Matrix of C(Q). (Expressions for their individual entries appear in section 3.3 and table 4.1 respectively while their expression in matrix form appears in table 4.2 .)

The Model/Trust-Region approach (for minimization) is based on computing a step that minimizes a mathematical model of the cost function within a region in which this model is trusted to be accurate enough.

Let Q_c be the current point and $Q_n = Q_c + q_n$ be the next point to be computed, the method assumes a quadratic model, for the cost function, of the form:

$$m(Q_n) = m(q_n) \equiv C(Q_c) + g^t(Q_c)\, q_n$$
$$+ (1/2)\, q^t{}_n\, H(Q_c)\, q_n$$

The method finds an approximate minimizer for the model within the Model/Trust-Region T defined as

$$T = \{\ Q \text{ in } S: \ ||D(Q - Q_c)|| \leq r\ \}$$

where D is a diagonal matrix used for scaling of the independent variables (see below) and $r \geq 0$ is the "trust radius" (below).

When the minimizing step q_n has been computed the value of the cost function at Q_n is evaluated using the model $m(Q_n)$. Thus the reduction in function value for the step determined can be predicted. Then the actual function evaluation $C(Q_n)$ is made and the function reduction achieved is compared to the predicted one. If agreement is "too good" (for accurate explanation see [G 80] or [G 83]) r is increased (and q_n determined again) since this fact is taken as a sign that the extent of the trust region is being underestimated. Similarly, if agreement is "bad" the value of r is decreased and q_n is evaluated again. For the detailed implementation of the step selection, including updating of the trust radius, see [G 80].

The Model/Trust-Region approach has the advantage that the steps chosen become full Quasi-Newton steps when regions of quadratic behaviour are entered, and the steps are close to Cauchy steps (steepest descent) when the quadratic model is not appropriate. It also has the advantage that, unlike line search methods to determine the lengths of the next step, it uses the full quadratic model to find a minimizer within the region T as oppose to finding the minimizer along a line of search (line search) by just using a one dimensional quadratic model along this line.

Subroutine humsol [G 80], coded by Gay, gives the user the option of providing the exact expression for the gradient and Hessian of the cost function. We have chosen it for our algorithm.

Scaling of the independent variables to roughly equalize their typical size has the effect of helping the algorithm to treat more evenly the variables in the problem. This improves the speed of convergence of the method in use. In humsol we selected the option of computing scaling parameters for the independent variables based on the diagonal entries of the Hessian matrix provided.

Scaling would be important in networks where there is great difference in the magnitudes of the distance measurements given.

A global method for minimization will converge, in general, to a local minimum. If this local minimum is not a global minimum the algorithm will typically have to be restarted elsewhere. Convergence to a local minimum can be detected by computing the norm of the gradient since at a local minimum its norm must be zero. In our case a global minimum will also have a cost function value of zero. (Of course, in a finite precision machine these values will only approach zero.) The stopping tests in humsol combine checks for the value of the norm of the gradient and checks for the relative reduction (with respect to the previous iteration) of the cost function value and of the gradient norm. Details for the implementation of stopping tests appear in [G 80]. We note however that these tests are invariant under scaling of the variables.

Scaling of the cost function is also helpful. In our case this can be done through the edge weights A_{ij} and B_{ij} of the cost functions. We have selected them so that edge errors are turned into percentage errors for each edge. Details of this scheme are given in the next section.

4.9 SOFTWARE DESCRIPTION AND COMPUTER OUTPUTS

Two programs were produced to implement the iterative method. Their current versions are called "euclides" and "plsyst". Program plsyst carries out the minimization procedure making use of subroutine humsol (see section 4.8). Two data files are used, the "input file" (input to plsyst) and an "output file" (from plsyt). The input file is generated by program euclides and the output file is used by plsyst to print the iteration data and results.

Program euclides is used to generate the input frame to the Position Location algorithm for experimentation. It obtains from the user (interactively) the underlying graph for the problem. Two options are provided. One allows the user to provide the positions of the vertices of the frame. In the second option the program generates the position of the vertices using a random number generator subroutine. A uniform probability distribution over the range [-10.0,+10.0] is used to generate the value of each coordinate of each vertex, unless the coordinate will be used (constrained) in the Grounding process. (The value assigned to coordinates involved in Grounding is zero, as explained below.) Program euclides then computes the distance set vector for the graph provided. This information is written on the input file in a format compatible with program plsyst. Program euclides then generates the initial realization of the frame for use by program plsyst. Again the two options above are available for this part. If the option using random number generation is chosen then the program sets some coordinates (see below) to zero as a convenient way to implement Grounding together with program plsyst. For a frame of n-dimensional Positions, program euclides assumes that the first n+1 vertices (as defined by the user) will be used as the Root for Grounding. For node i, i=1,...,n+1, the program assigns zero to all but the i-1 first coordinates (defined by the user). These coordinates will be used by program plsyst for Grounding. The user is responsible for implementing this process if the first option (user provided coordinates) is chosen instead.

Program plsyst reads the input frame from the input file and calls subroutine humsol to perform the minimization on the problem. The user determines the number of dimensions for the space of operation and the type of cost function (A or B) to be used in the iterations. In the current version, plsyst assigns the weights for the cost functions as

$$A_{ij} = 1/(||D_{ij}||^4)$$

$$B_{ij} = 1/(||D_{ij}||^2)$$

and the normalizing constants p_A and p_B as

$$p_A = 1/q$$

$$p_B = 1/q$$

where variable "q" is the number of edges of the graph. The choice of weights turns the error contributed by the edges into errors relative to their true length (the distances given). The value of the scaling variable "p" is chosen to give the cost functions a value of 1.0 when each of the edge vectors in the realization evaluated produces an edge error of 100 % with respect to the square of their true length (type A, see equation 3.3.a) or with respect to just their true length (type B, see equation 3.3.b).

Program plsyst contains the three routines required by humsol that compute the value of the cost function, the gradient and the Hessian matrix for a given realization. The cost function value is evaluated according to equations 3.3. The gradient is evaluated according to equations 3.4 and the Hessian is evaluated using the expressions in Table 4.1.

Grounding is implemented in the program by leaving the coordinates involved in the Grounding process unaltered through the algorithm. The coordinates are selected as in program euclides. These coordinates are not considered variables in the minimization process and unconstrained minimization is carried out on the rest of the coordinates which take no part in Grounding. The minimization procedure is carried out by subroutine humsol, including printing of the iteration data. Program plsyst allows the user to select the input (control) parameters for subroutine humsol. Any non-default parameters selected are printed by humsol on the output file. (For the results presented here we have chosen the default parameters, except for "absolute function convergence tolerance", AFCTOL, which was set to 0.0025.)

None of the two programs described contains provisions for checking the rigidity of the frame either by computing the Jacobian matrix of the measurement function or by checking the connectivity properties of the graph. In this respect we have to follow a trial an error process for frames where rigidity is not evident. However any frame, rigid or not, is useful for testing the method.

We have not implemented either a check to verify automatically if the vertices of the Root used form a Maximally Spanning set of vertices. No graphic facilities have been implemented for displaying in real time the position of the vertices through the iteration process either.

These last three features mentioned would be useful in facilitating analysis of the experiments.

In the pages that follow we display some of the output data obtained using the software described above. We have iterated on input frames which are rigid. A case like the "irregular_hexagone" (complete rigidity) seems to provide convergence from different starting realizations. Even so there are instances where the initial realization presents problems (see irregular_hexagone3). Cases like "Tutte10_A" seem to correspond to a rigid frame with several positions. In this cases we observe convergence to dissimilar configurations yet the edge errors seem small enough at the time the algorithm stops to suggest that we are arriving at valid Positions of the frame. A case like "partition_8" seems to illustrate a clear case of conflict point.

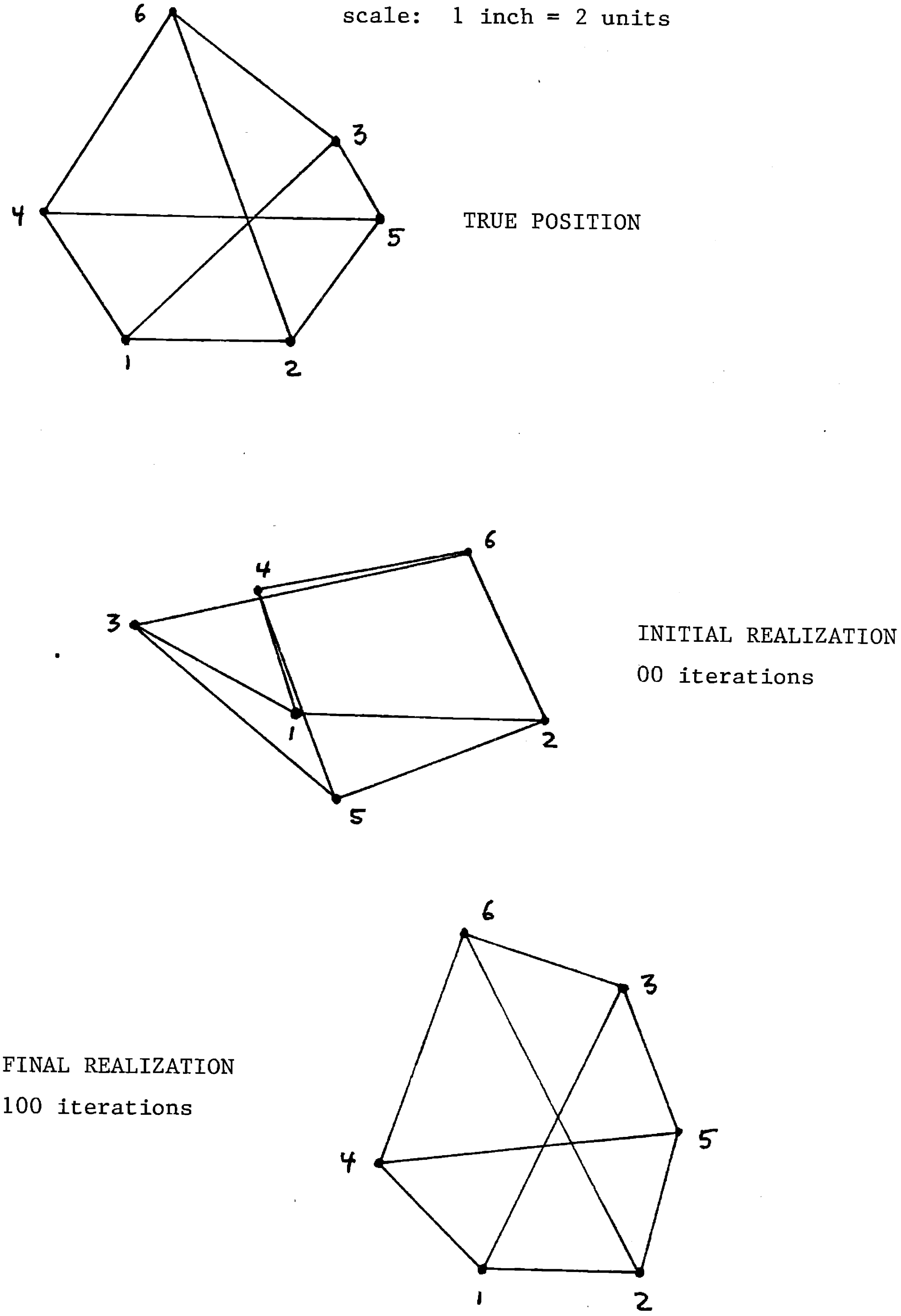
More experience in using the software would be required to take better advantage of the minimization routine. The number of iterations recorded seem unnecessarily high. This may be suboptimal use of the input parameters to subroutine humsol.

Finally, the results reported were obtained using the type B cost function. Use of the type A cost function seems to yield overall similar results, but we have not tested it as much as type B.

Figure 4.3: irregular_hexagone

comment: rigid frame.

scale: 1 inch = 2 units



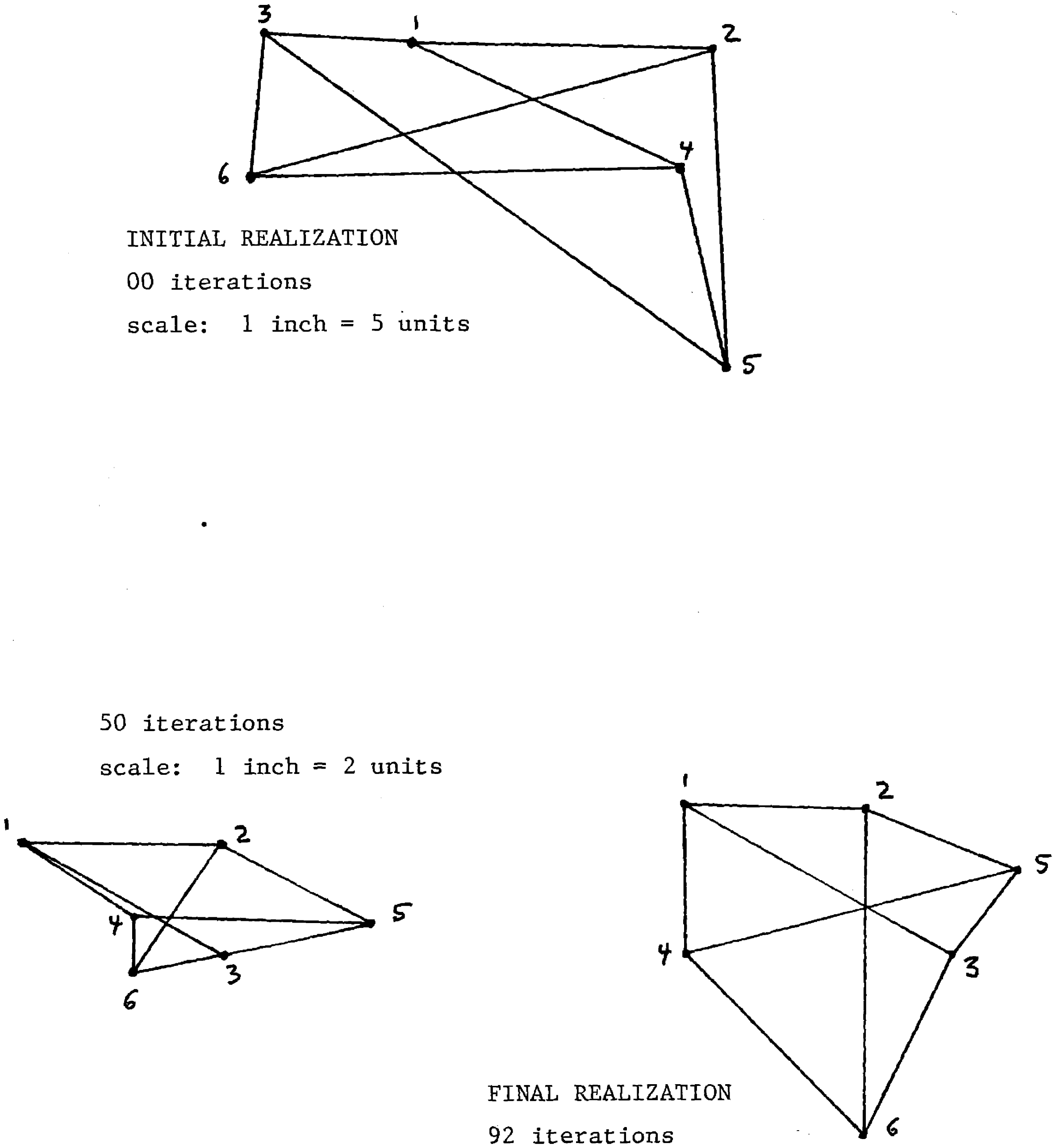
TRUE POSITION

INITIAL REALIZATION
00 iterations

FINAL REALIZATION
100 iterations

Figure 4.4: hexagone_A04

comment: input frame as for irregular_hexagone.



INITIAL REALIZATION

00 iterations

scale: 1 inch = 5 units

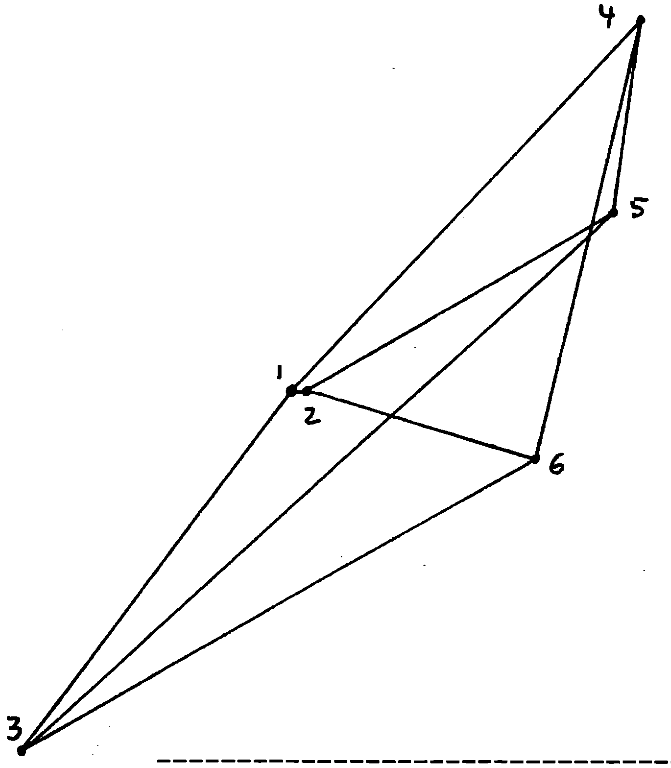
50 iterations

scale: 1 inch = 2 units

FINAL REALIZATION

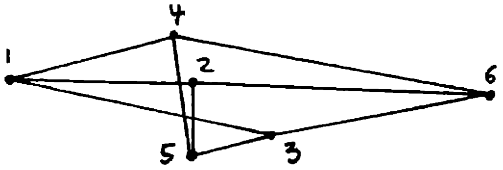
92 iterations

Figure 4.5: hexagone_A05
comment: input frame as for irregular_hexagone.

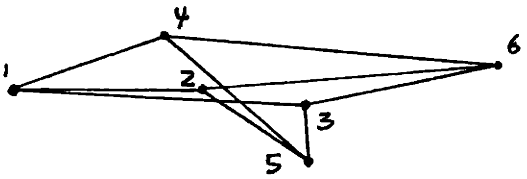


INITIAL REALIZATION
00 iterations
scale: 1 inch = 5 units

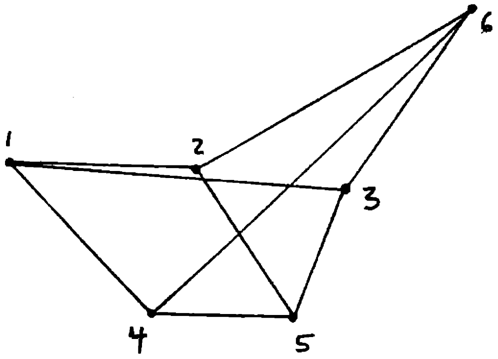
scale: 1 inch = 2 units



50 iterations



100 iterations

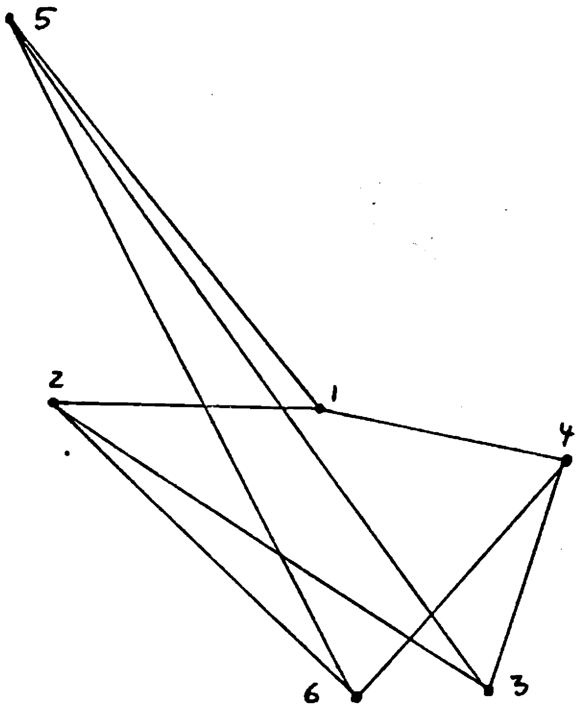


FINAL REALIZATION
150 iterations
Iteration Limit. Maximum absolute value of edge error (pcnt.) = 25%

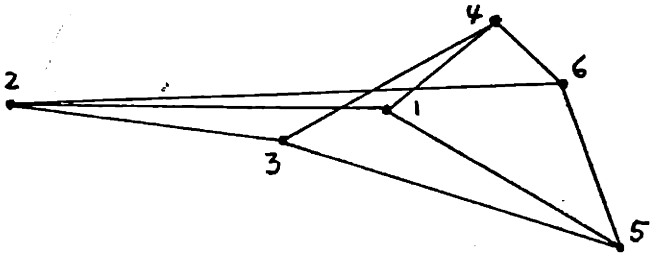
Figure 4.6: irregular_hexagone3

comment: topology as for irregular_hexagone.

scale: 1 inch = 5 units

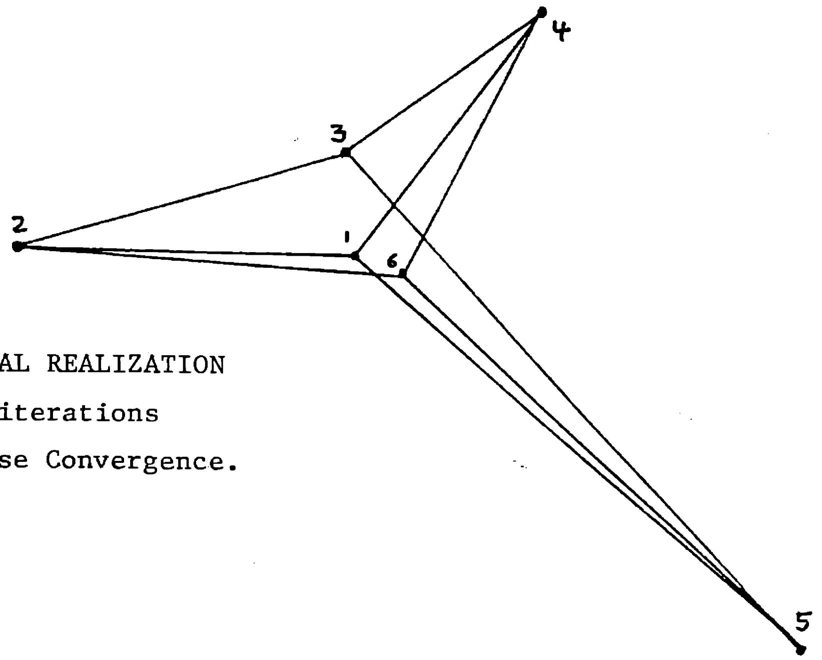


TRUE POSITION



INITIAL REALIZATION

00 iterations



FINAL REALIZATION

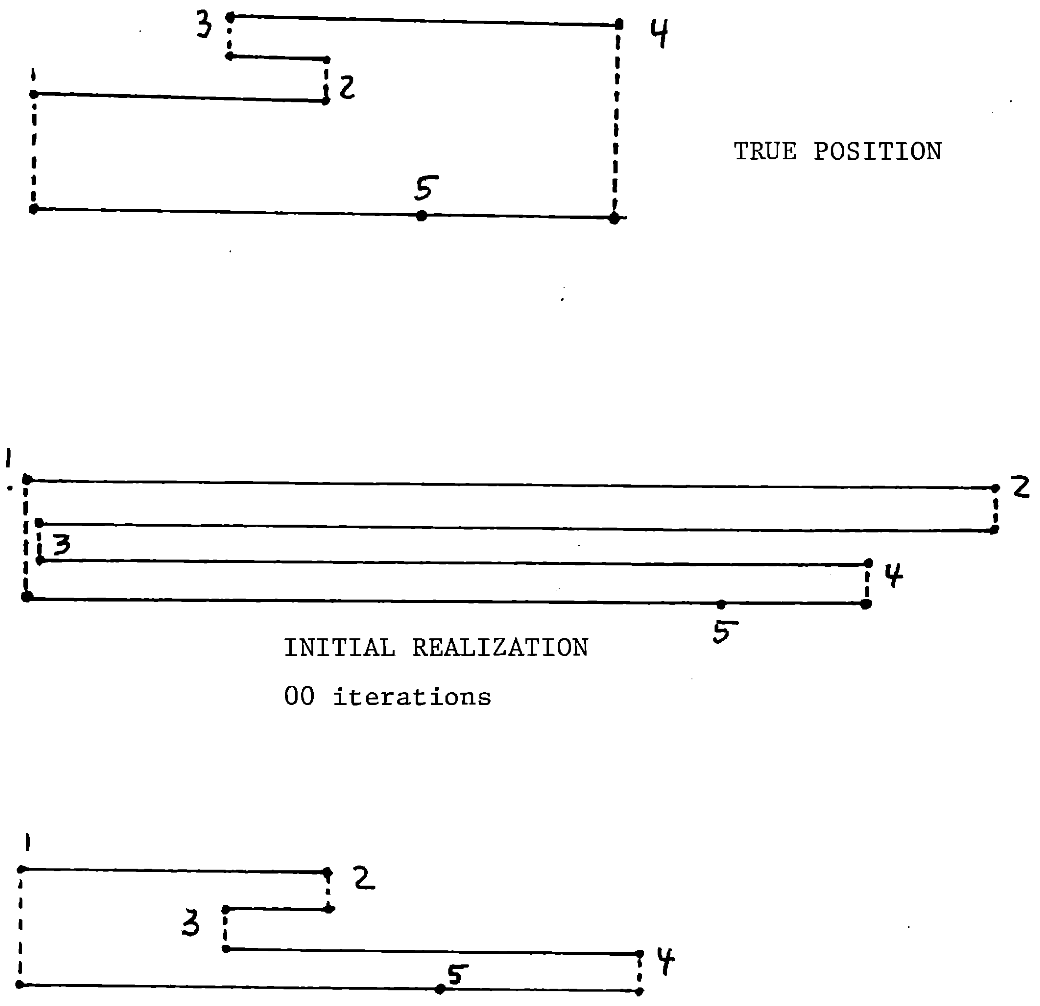
61 iterations

False Convergence.

Figure 4.7 : partition_5

comment: 1-dimensional realizations.

scale: 1 inch = 2 units



TRUE POSITION

INITIAL REALIZATION
00 iterations

FINAL REALIZATION
45 iterations

Figure 4.8: partition_8

comment: 1-dimensional realizations.

scale: 1 inch = 2 units

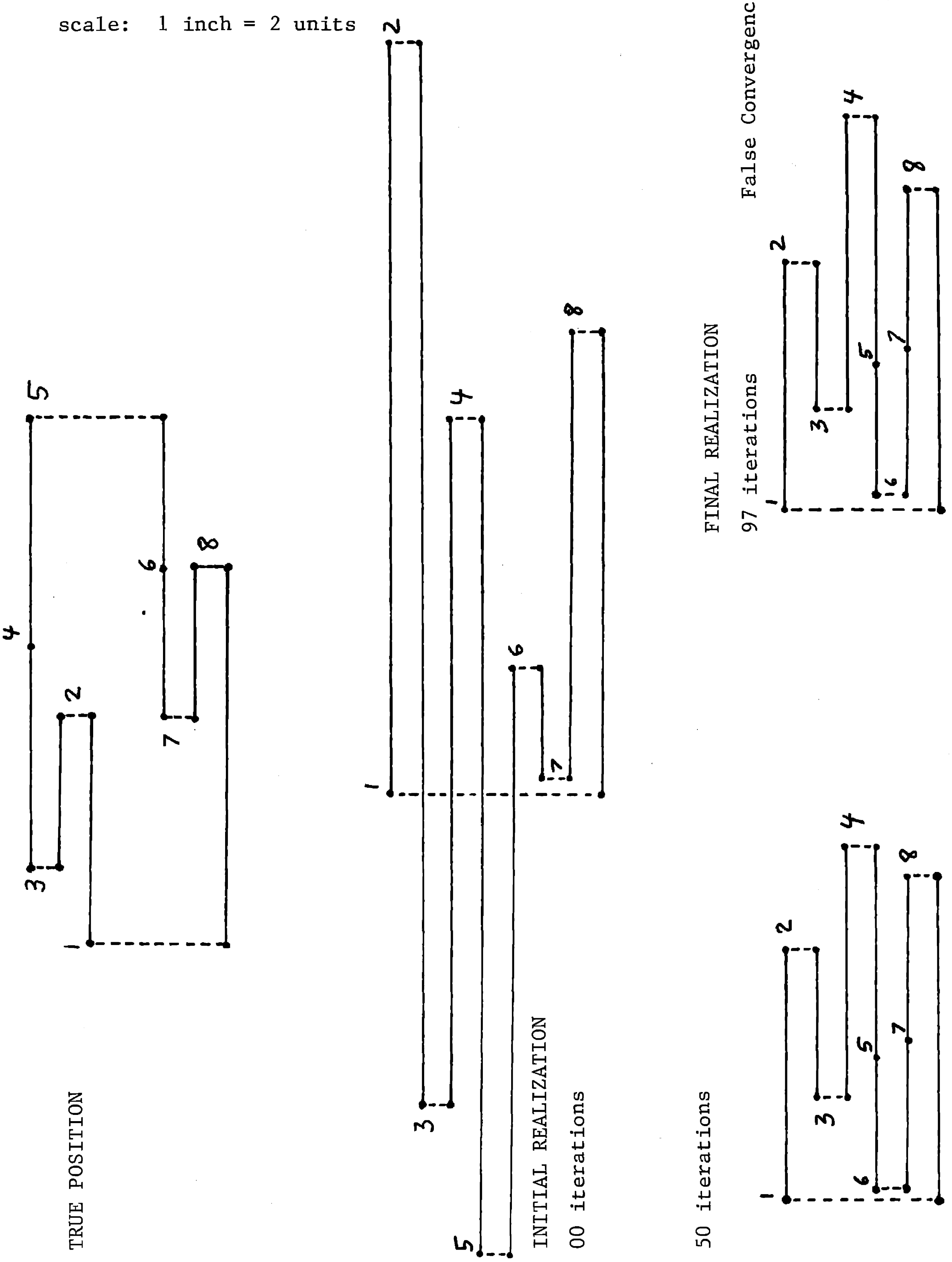
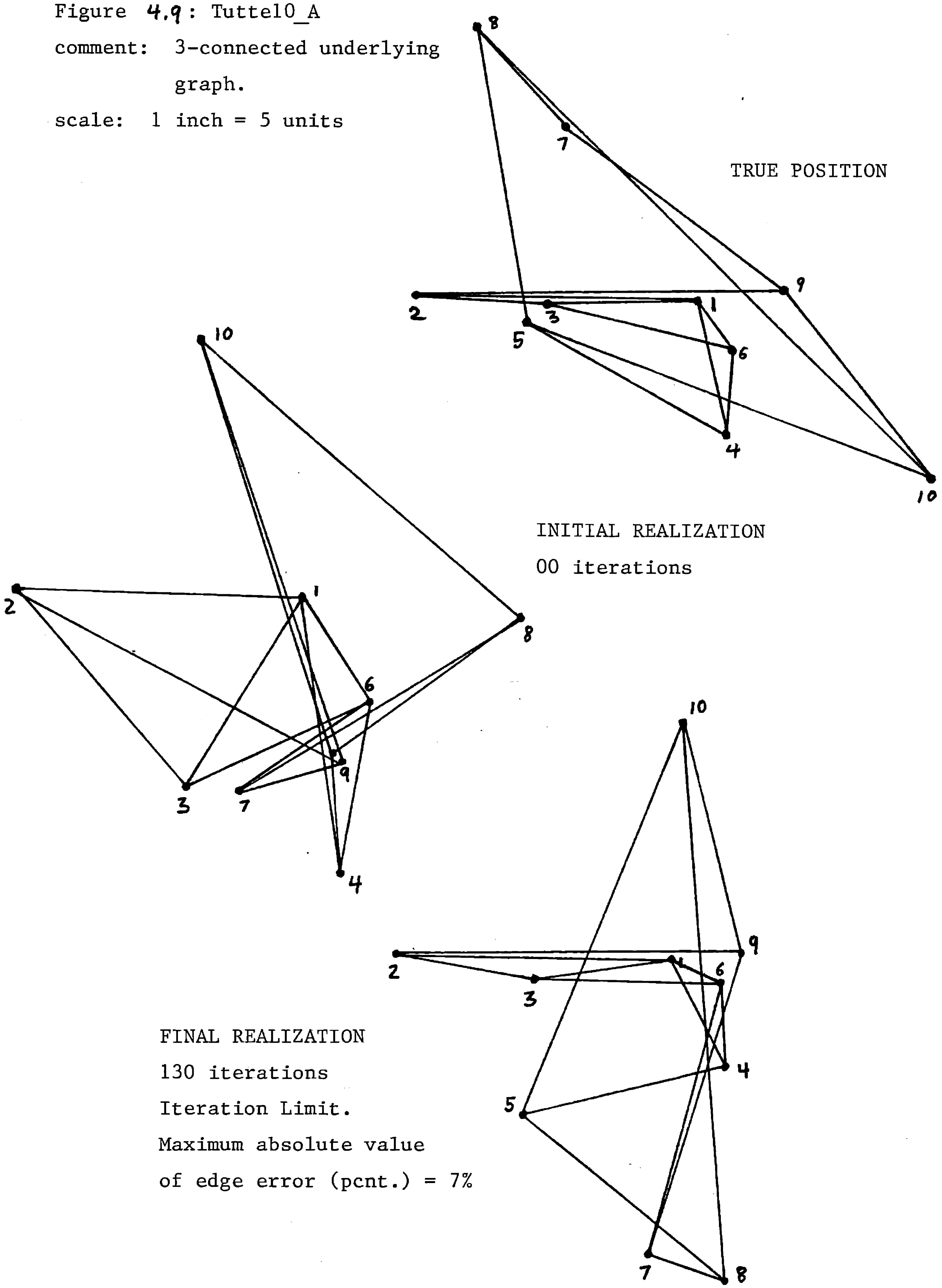


Figure 4.9: Tuttel0_A
comment: 3-connected underlying graph.
scale: 1 inch = 5 units


TRUE POSITION

INITIAL REALIZATION
00 iterations

FINAL REALIZATION
130 iterations
Iteration Limit.
Maximum absolute value of edge error (pcnt.) = 7%

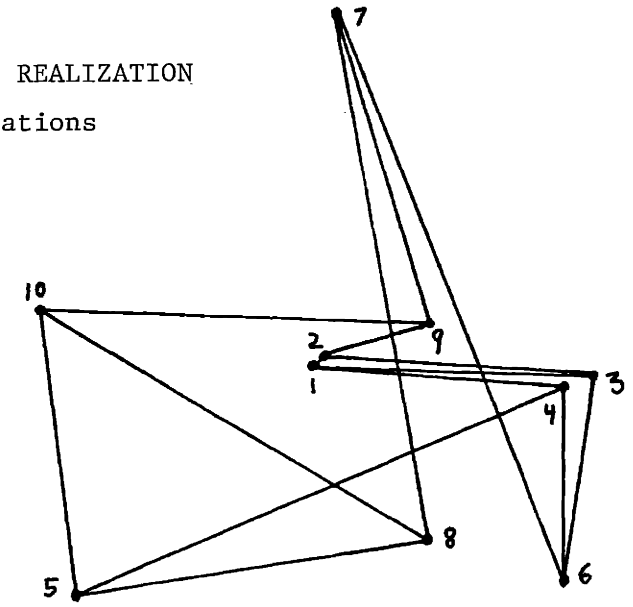
Figure 4.10: Tuttel0_A02

comment: 3-connected underlying graph. Input frame as for Tuttel0_A.

scale: 1 inch = 5 units

INITIAL REALIZATION

00 iterations



FINAL REALIZATION

150 iterations

Iteration Limit.

Maximum absolute value of edge error (pcnt.) = 30%

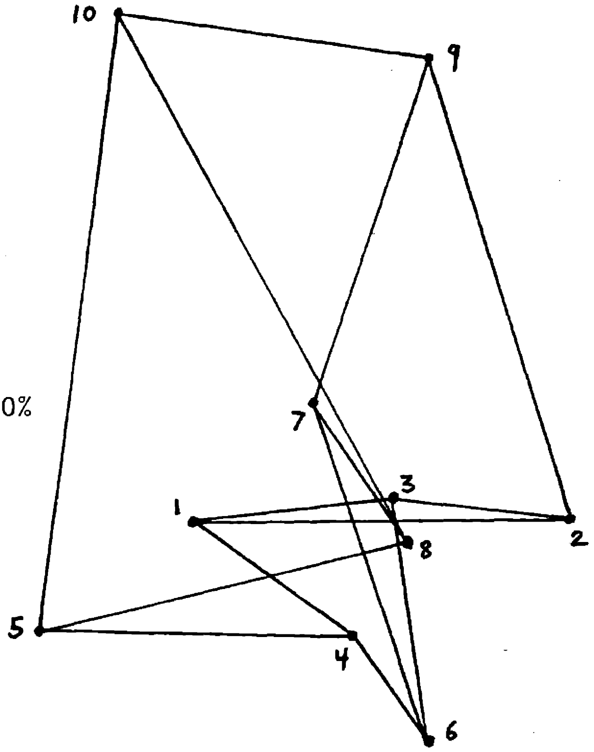
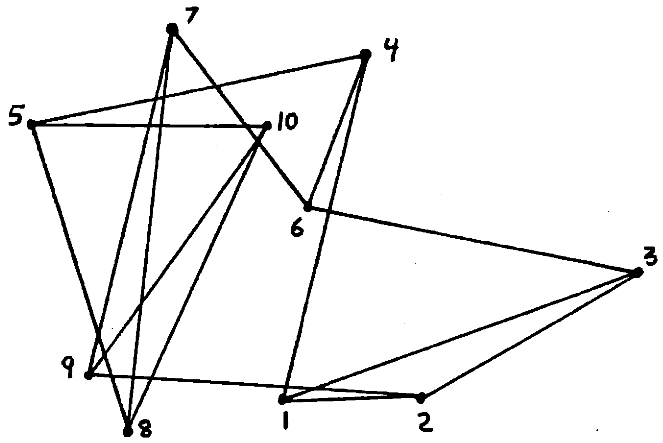


Figure 4.11: Tuttel0_A03

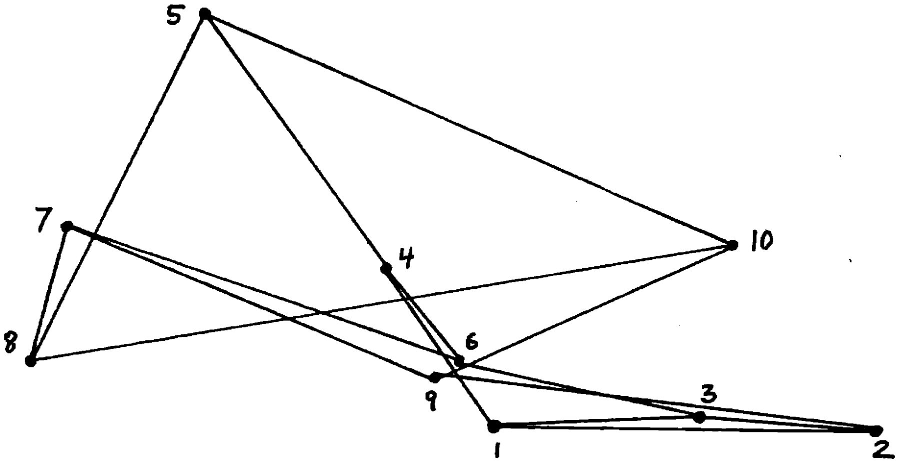
comment: 3-connected underlying graph. Input frame as for Tuttel0_A.

scale: 1 inch = 5 units



INITIAL REALIZATION

00 iterations



FINAL REALIZATION

66 iterations

Absolute Function Convergence.

CHAPTER 5

In section 2.3 we briefly mentioned the problem of measurement errors. They are likely to make a PL algorithm converge to an answer different from the one that would have been obtained if exact measurements were available. The modified realization could be regarded then as the original realization plus an error portion induced by the measurement errors. The purpose of this chapter is to discuss the relationship between these measurement errors and the errors they induce in the original realization.

5.1 THE PROBLEM OF MEASUREMENT ERRORS

Having a PRN scenario in mind it is to be expected that the distance measurement process will not be completely accurate. For the same underlying graph representing the distance measurements available we will speak of an "exact" distance vector and a "modified" distance vector. By exact distance set we refer to a set produced by a perfectly accurate measurement process. By a modified distance vector we refer to that produced by a process which may contain measurement errors. Similarly we will use the terms exact and modified referring to frames and Positions of a frame.

Two possible situations exist when we have a modified distance vector.

A- There exists a realization of the vertices which is compatible with this modified distance vector. Then the underlying graph and this modified distance vector give rise to a "modified frame" whose Positions can be regarded as "modified Positions" of the Positions for the exact frame.

B- There does not exist a realization compatible with this modified distance vector. Hence, since the definition of frame (section 2.1) implies the existence of a compatible realization, the underlying graph and this modified distance vector do not form a frame proper. The ideal PL problem, as defined in section 2.3, does not have a valid input (a frame). However, we could feed this inexact information to the PL algorithm and allow it to iterate. We will refer to the available information in this case as a "pseudo-frame" and will consider it as a valid input to the PL algorithm. Assuming the algorithm converges, it will arrive at a given realization for which there will be some corresponding frame. The value of the cost function for this realization however will be greater than zero. We can still consider this frame as the modified frame induced by the measurement errors.

Thus, both cases above are equivalent to exact PL problems where the input is the modified frame instead of the exact frame.

We can represent the errors of measurement by a vector just as we represent the distance measurements themselves by a distance vector.

For the analysis we assume that we have an algorithm capable of finding the correct solutions for the PL problem in the exact case. Consider now a PL problem where measurement errors are present. Let the exact case, i.e. no errors present, correspond to a rigid input frame $T_m(G,D)$ where D is the exact distance vector. In the modified case the distance set available will correspond to some modified vector D'. We define the "distance error vector" d as

$$d \equiv D' - D \qquad 5.1$$

If the answer of the PL algorithm to the exact case were denoted by Q (a realization vector) and the answer produced in the modified case is denoted by Q^* then we define the "position error vector" q as

$$q \equiv Q^* - Q \qquad 5.2$$

Q is compatible with D and the exact frame $T_m(G,D)$. On the other hand, Q^* will be compatible with some exact input frame $T_m(G,D^*)$.

If the input to the algorithm is a frame then $D^* = D'$. If the input to the algorithm is a pseudo-frame $D^* \neq D'$.

The problem of measurement errors consists of determining the relationship between the distance error vector "d" and the position error vector "q" induced by it. We will consider in particular the case when the measurement errors are small in magnitude as will be explained in section 5.3. Section 5.2 contains definitions and equations necessary for section 5.3.

5.2 JACOBIAN MATRICES OF INTEREST

Allowing for the presence of measurement errors we consider the cost functions to be functions of both the position of vertices in the space of operation S^+ and of the distance measurements available. We denote the cost functions by C(Q,D) to emphasize this dependence; D is the distance vector. Similarly, all the matrices related to the cost function derived in previous chapters (e.g. gradient, Hessian, Jacobian) are also considered functions of Q and D.

For a given realization vector Q and measurement function $M_G(Q)$ it will be convenient to recall or define the following matrices.

H(Q,D) = Hessian matrix of the cost function (c.f. chapter 4).

g(Q,D) = gradient of the cost function (c.f. chapter 3).

$J_M(Q)$ = Jacobian matrix, with respect to Q, of $M_G(Q)$.

$J_Q(Q,D)$ = Jacobian matrix, w.r.t. Q, of gradient g(Q,D).
It is the matrix of second order derivatives H(Q,D).

$J_D(Q,D)$ = Jacobian matrix, w.r.t. D, of gradient g(Q,D).
It is also a matrix of second order derivatives.

It is important to note the differences between this three matrices to avoid confusion. $J_M(Q)$ is obtained from a "measurement function" and the derivatives are with respect to the positions of the vertices. $J_Q(Q,D)$ is obtained instead from the gradient of the cost function with respect to the positions of the vertices. Finally, $J_D(Q,D)$ is also obtained from the gradient of the cost function but derivatives are now with respect to the distances in the distance set.

Unfortunately the subscript notation in these matrices is not self-consistent; we have adopted it for simplicity.

The expression for $J_M(Q)$ has already been derived (section 4.1). If we now take the gradient of the cost function (section 3.3) and want its Jacobian matrix w.r.t. Q we obtain simply

$$J_Q(Q,D) = H(Q,D) \qquad 5.3$$

For the last Jacobian matrix we have (c.f. equation 4.3)

$$J_D(Q,D) = \begin{bmatrix} \\ \text{---}\ \mathrm{grad}^t\ g_{ix}(Q,D)\ \text{w.r.t.}\ D\ \text{---} \\ \\ \\ \end{bmatrix} \qquad 5.4$$

We now derive explicit expressions for the components of the matrix $J_D(Q,D)$. For simplicity we will denote the matrix $J_D(Q,D)$ for cost functions A and B respectively as $J^A{}_D$ and $J^B{}_D$; similarly for the matrix $J_M(Q)$. u_{ij} is the unit vector in the direction r_{ij}.

By making use of the results in Table 4.2 for the gradient of the cost functions and for the expressions of a Jacobian we obtain:

$$J^A{}_D(Q) = J_D(g^A) \qquad 5.5.a$$

$$J^A{}_D(Q) = -(8/p_A)\ J^t{}_M(Q)\ R\ A\ F \qquad 5.6.a$$

$$J^B{}_D(Q) = J_D(g^B) \qquad 5.5.b$$

$$J^B{}_D(Q) = -(2/p_B)\ J^t{}_M(Q)\ B \qquad 5.6.b$$

Equation 5.4 can then be expressed, for each case, as:

$$J^{A}{}_{D} = -p^{A} \, (J_{M})^{t} \, W^{A} \qquad 5.7.a$$

$$J^{B}{}_{D} = -p^{B} \, (J_{M})^{t} \, W^{B} \qquad 5.7.b$$

where $p^{A} = (8/p_{A})$, $p^{B} = (2/p_{B})$ and the matrices W^{A} and W^{B} are diagonal matrices of dimensions e by e and diagonal components "(i,j)" given by:

$$W^{A}{}_{ij} = A_{ij} \, D_{ij} \, r_{ij} \qquad 5.8.a$$

$$W^{B}{}_{ij} = B_{ij} \qquad 5.8.b$$

In both cases then we have the general equation

$$J_{D}(Q,D) = -p \, J_{M}{}^{t} \, W \qquad 5.9$$

5.3 SMALL ERRORS APPROXIMATION

Consider now an exact PL problem. Let the input be a frame with exact distance vector D and let the realization vector Q be the answer of the PL algorithm. Let a modified distance vector be D + d and the answer for the modified case be Q + q. Then the gradient vector g(Q,D) of the cost function for the exact case will be modified to a vector g(Q+q,D+d). If we assume that both q and d are small enough so that we can apply Taylor's theorem around the exact solution we have to a first order approximation:

$$g(Q+q,D+d)= g(Q,D) + J_{Q}(Q,D) \, q + J_{D}(Q,D) \, d \qquad 5.10$$

Since Q is a solution for the exact case and Q+q is a solution for the modified case, it follows that

$$g(Q+q,D+d) = g(Q,D) = 0\text{-vector}$$

and replacing into 5.10

$$J_Q(Q,D)\ q = -\ J_D(Q,D)\ d$$

hence $$H(Q,D)\ q = -\ J_D(Q,D)\ d \qquad 5.11$$

where the last step follows from equation 5.3.

For simplicity denote $J_M(Q)$ by J. Recall that at the solution Q the Hessian matrix is given by (equation 4.22)

$$H(Q,D) = p\ (J^t\ T\ J) \qquad 4.22$$

since all edge errors are equal to zero.

So, replacing 4.22 and 5.9 into 5.11

$$J^t\ T\ J\ \ q = J^t\ W\ d \qquad 5.12$$

which relates the position error vector q to the distance error vector d, as desired. On the assumption that the distance errors and the position errors induced are suitably small it is reasonable to assume that in the above relationship (5.12) we can approximate the Jacobian matrix J, which refers to a realization compatible to the original exact frame (and hence is not available to us), by the Jacobian matrix $J_M(Q+q)$ which can be computed from the realization obtained through the algorithm.

5.3.1 grounded case.

Consider now the case were the frame is infinitesimally rigid and grounded. Recall that we can write (c.f. section 4.7)

$$J^t \; T \; J = (T^{1/2} \; J)^t \; (T^{1/2} \; J)$$

The null space of $J^t \; T \; J$ is the same as that of J since $T^{1/2}$ is invertible (c.f. equation 5.14). Since for the infinitesimally rigid and grounded case the null space of J is empty (c.f. section 4.5) then $J^t \; T \; J$ is invertible. Thus for the infinitesimally rigid and grounded case we obtain:

$$q = (J^t \; T \; J \;)^{-1} \; J^t \; W \; d \qquad 5.13$$

which gives the position error vector as a straightforward linear function of the distance error vector.

Note that, from equation 4.20, the diagonal elements "(i,j)" of T are found to be

$$T^A{}_{ij} = A_{ij} \; D_{ij}{}^2 \qquad 5.14.a$$

$$T^B{}_{ij} = B_{ij} \qquad 5.14.b$$

since $||r_{ij}|| = D_{ij}$ at Q.

It may be of interest in a PL problem to estimate how large an error is introduced in the answer of the algorithm due to measurement errors. Thus suppose that we have an estimate of the measurement errors in the system and use the PL algorithm to arrive at an answer. The interesting case arises when measurement errors are not large enough so as to render the answer totally useless. In a case where the errors are small we could estimate the errors induced in the position of the vertices by using equation 5.13.

The framework developed for measurement errors may also be used, for example, to investigate the effect of round off error in the distance information. Rounding off of the distance measurements could be represented by a suitable distance error vector.

Equation 5.13 also reveals explicitly the strong dependence of the resulting position errors on both the type of cost function chosen and the realization found as the solution (the realization shows up through the matrix J). This means for example that the position error vector is in fact dependent on the vertices chosen for grounding since they determine the location of the compatible realization Q that will be obtained as an answer.

5.3.2 Non-grounded case.

Finally, if we consider the case of an infinitesimally rigid frame which is not grounded the matrix J^t T J would not be invertible since the null space of J is non-empty. The vector q in equation 5.12 would not be unique. For any q that satisfies 5.12, the vector $q + q^0$, where q^0 is a vector corresponding to any trivial displacement, will also satisfy equation 5.12.

Because the frame is not grounded this is to be expected since the situation is equivalent to having a realization $Q_1 = Q + q^0$ as the answer for the exact case (which is a valid answer since q^0 denotes a trivial displacement) and a modified realization $Q_1{}'$ + q as the answer for the modified case. The situation will be discussed in greater detail below.

5.4 ANOTHER APPROACH: BEST APPROXIMATION EQUATIONS

Equation 5.12 can be written, for the Type A cost function, as

$$[(T^A)^{1/2}\ J]^t\ [\ (T^A)^{1/2}\ J]\ q = [(T^A)^{1/2}\ J]^t\ (T^A)^{1/2}\ R\ F\ d$$

or

$$(M^A)^t\ M^A\ q = (M^A)^t\ v^A \qquad 5.15.a$$

where matrices R and F are as in section 4.6, matrix $M^A = (T^A)^{1/2}\ J$ and v^A is a vector whose components corresponding to a given edge "(i,j)" are given by

$$v^A{}_{ij} = (A_{ij})^{1/2}\ D_{ij}{}^2\ d_{ij} \qquad 5.16.a$$

where d_{ij} is the corresponding component of the vector d.

for the Type B cost function:

$$[(T^B)^{1/2}\ J]^t\ [\ (T^B)^{1/2}\ J]\ q = [(T^B)^{1/2}\ J]^t\ (T^B)^{1/2}\ d$$

or

$$(M^B)^t\ M^B\ q = (M^B)^t\ v^B \qquad 5.15.b$$

where $M^B = (T^B)^{1/2}\ J$ and $v^B = (B_{ij})^{1/2}\ d_{ij}$

Generalizing for types A and B:

$$M^t\ M\ q = M^t\ v \qquad 5.17$$

Equation 5.17 corresponds, in linear algebra terms, to the equation that defines the "Best Approximation" solution q to an equation ([S])

$M\ x = v$ where x is a vector of unknowns

This last equation may or may not have a solution, and the Best Approximation solution is defined to be a value q of the vector x such that the norm of the difference vector M x - v is a minimum. Such a value satisfies equation 5.17.

Now consider the following special case:

Let $A_{ij} = (D_{ij})^{-4}$ for all (i,j)

and $B_{ij} = 1$ for all (i,j)

Then, replacing into 5.14,

$$J^t\ J\ q = J\ v \qquad 5.18$$

which is the equation for the Best Approximation solution to the equation

$$J\ x = v$$

The expression for the Type B cost function is of special interest. For this type, equation 5.18 becomes the Best Approximation equation for

$$J\ x = d \qquad 5.19$$

This equation has an intuitive interpretation. Consider again a frame which is infinitesimally rigid (grounded or otherwise). In the presence of measurement errors the algorithm will try to achieve a realization Q^* compatible with the modified distance set D' by altering the exact realization vector Q by q (the position error vector).

From the interpretation of the range of the gradient (section 4.2), an alteration q in the realization will produce an alteration J q in the measurement function, so the algorithm leads to the position error vector x = q that minimizes the euclidean norm $||d - Jx||$, which is the interpretation of a Best Approximation solution. i.e. q is the vector such that J q will best approximate d.

5.5 MINIMUM NORM BEST APPROXIMATION SOLUTION

Consider the following matrix equation:

$$A \ x = y \qquad 5.20$$

where the null space of matrix A is non-empty, y is a known vector and x is a vector of unknowns.

Let x_0 be a solution to this equation and u be a vector in the null space of A. Then

$$A\ (x_0+u) = A\ x' = y$$

Many values of x' satisfy this matrix equation.

The minimum norm solution to the equation above is defined to be a vector x_{MN} such that

$$||x_{MN}|| \leq ||x||$$

among all x which satisfy the equation.

The minimum norm solution satisfies the following equation ([C]):

$$A\ x = A\ A^t\ y$$

It is a vector orthogonal to the null space of the matrix A and any vector satisfying 5.20 can be expressed as the sum $x_{MN} + u$ for some value of u.

We consider now the relationship between the position errors induced by small distance errors in the case when Grounding is applied and the position errors induced when no Grounding is applied.

Consider an infinitesimally rigid input frame for the PL algorithm. When Grounding is applied the realization obtained by the algorithm wil be some realization

$$Q_0^+ + x^+$$

where Q_0^+ denotes the realization that would be obtained for the "exact" frame and x^+ is the position error induced by the distance error vector, d say. We use the superscript "+" to emphasize that the position error vector has a reduced number of dimensions because of the use of Grounding. It is reasonable to assume that for suitably small values of d the induced position error vector x^+ is small and hence satisfies equation 5.12 (and hence 5.13 since the frame is grounded). On account of Grounding the Jacobian matrix in these equations is a matrix with reduced structure since the whole function space has been reduced in dimensions to suppress trivial displacements of the frame.

To the vectors Q_0^+ and x^+ there correspond vectors Q_0 and x obtainable by augmenting the dimensionality of the former two vectors until they have the full set of variables in the non-grounded system. Components of the vectors corresponding to grounded variables will simply have some constant value (e.g. zero). The augmented vectors will satisfy equation 5.12 with matrix J in the expression given by $J_M(Q_0)$.

In the case when no Grounding is applied we could in general express the realization obtained by the algorithm in the form

$$Q_0 + q_T + z = Q_1 + z$$

where Q_0 is the same realization as in the grounded case, q_T is a vector corresponding to a trivial displacement of Q_0 and z is the position error vector, measured from Q_1, induced by the distance errors.

If z is assumed small enough, z satisfies equation 5.12 and matrix J in the equation is given by $J_M(Q_1)$.

Now, equation 5.12 is invariant up to trivial displacements of the realization Q that determines the Jacobian matrix $J_M(Q)$ in the expression for 5.12. This is the case since 5.12 defines the best approximation solution to an equation of the type

$$J_M(Q)\ q = \text{constant-vector} \qquad 5.21$$

which, up to trivial displacements, will only depend on the relative positions of the vertices in the realization.

Hence vectors x and z, for the Grounded and non-Grounded case respectively, satisfy the same equation up to trivial displacements.

The value of x is fixed and determined by the Grounding. If, on the other hand we imagined choosing the value of q_T, in the non-grounded case, that minimizes the norm of vector z the resulting value, z_{MNBA} say, would correspond to the minimum norm solution to equation 5.12. Alternatively, z_{MNBA} is the "minimum norm best approximation" solution to equation 5.21.

Hence the minimum norm solution to equation 5.12 is related to the augmented position error vector of the grounded case by:

$$z_{MNBA} = x + u_T$$

where u is a trivial displacement.

The minimum norm solution z_{MNBA} has the advantage of not being dependent on the orientation or location of the realization under iteration or on the choice of Root for Grounding.

5.6 STATISTICAL INTERPRETATION OF THE COST FUNCTION

In a PL problem we try to arrive at a realization compatible with the input frame based on its distance vector.

With measurement errors present the modified distance vector $D'(Q)$ is a function of the realization Q corresponding to the exact input frame and is also a function of the distance error vector d.

$$D'(Q) = D(Q) + d$$

In this discussion we now consider to be a random vector with some statistical characterization.

If a measurement process results in a modified distance vector D_0' the aim of Position Location is to determine the value of the vector Q based on the vector D_0'. In this respect Position Location can be treated as an estimation problem.

With no a priori information on the configuration of the frame we may consider Q as a parameter to be estimated and seek the Maximum Likelihood estimate of Q, i.e. the value of Q that maximizes the conditional probability density function of $D'(Q)$ given Q for the value $D'(Q) = D_0'$ obtained. (Note that by probability density function of the vector $D'(Q)$ we mean the joint probability density function of the vector components $D'_{ij}(Q)$.)

Let us assume that the errors in the distance measurements can be modelled as a set of zero mean gaussian random variables with covariance matrix "K". Given Q, $D'(Q)$ is a gaussian vector of mean D(Q). Let k denote the determinant of matrix K. Let L be the number of distance measurements available. The conditional probability density function of $D'(Q)$ given Q is then,

$$f_{D'|Q}(D'|Q) =$$
$$(2\pi k)^{-L/2}\exp[(-1/2k)(D'(Q)-D(Q))^t\ K^{-1}\ (D'(Q)-D(Q))]$$

For $D'(Q) = D_0'$ the density function above is a maximum for values of Q such that the expression

$$(D'(Q)-D(Q))^t\ K^{-1}\ (D'(Q)-D(Q))$$

is a minimum. Now, recall that for cost functions of type B the PL algorithm seeks to minimize (c.f. section 4.6)

$$(M_G-D)^t \; B \; (M_G-D)$$

where M_G stands for the measurement function of the current realization in the algorithm and B is a diagonal matrix of edge weights. But, in the case where the measurement errors form a set of mutually independent gaussian random variables K^{-1} is also a diagonal matrix and this last expression is precisely our expression above if B equals K^{-1}. Hence, with this choice of matrix B, if convergence is achieved the realization found will be a Maximum Likelihood estimate for the case considered (independent zero mean gaussian errors with covariance matrix K).

CHAPTER 6

6.1 SUMMARY

We have given one formal statement of the Position Location Problem in Packet Radio Networks. We consider the distance measurements available and the graph underlying the problem as the input information to the problem. We refer to this information as "frame" (chapter 2) or "pseudo-frame" (chapter 5) depending on whether the distance measurements available form a consistent set of measurements or an inconsistent one. The problem asks if it is possible to find the relative positions of the nodes of the network represented by the frame and if so to find these relative positions. We defined the terms "rigidity" and "complete rigidity" of a frame (section 2.4). The question of finding relative positions translates then into the question: is the input frame completely rigid ?. In such a case the relative positions would be unambiguous. If the frame is rigid but not completely rigid the relative positions to be found for the nodes are ambiguous, since there would be many (countable) configurations for the network that still satisfy the information provided. Because there is no satisfactory characterization for complete rigidity so far, we have included a statement of the Position Location Problem where rigid frames which are not completely rigid are considered (relaxed version of the problem, section 2.3). In this last case the problem asks for any of the possible configurations of the network.

We described in general terms the combinatorial approach followed by the Incremental Position Location System (IPLS) proposed by Y. Yemini (section 1.4) and argue why an optimization approach to the problem may be an advantage. For this optimization approach we defined two types of cost functions and derived their gradient and Hessian matrices (chapters 3 and 4). These matrices for the cost functions defined show similar interesting structures which were used for analysis. The cost functions are defined so that their global minima correspond to solutions of the Position Location Problem.

The optimization approach was implemented and run on a few networks. Finally an analysis of the effect of relatively small errors on the distance measurements given is also presented.

6.2 CONCLUSIONS

In chapter 1 the problem of determining the existence of a solution for the Position Location problem is shown (after Yemini) to be NP-complete by showing how the PARTITION problem (NP-complete) transforms into the problem of existence of a solution.

We argue that an optimization approach to the PL problem avoids the combinatorial nature of the approach in IPLS. We do not circumvent the NP-complete nature of the problem of course, merely we have tried to avoid a combinatorial representation of the knowledge about the network. In this approach a cost function is assigned to the problem so that its (global) minimum value corresponds to valid configurations of the network. The problem turns into a non-linear unconstrained minimization problem. The minimization method chosen is explained in section 4.8.

We have included also results available in the literature on graph theoretical results related to rigidity of a frame. Of special importance are theorem 2.1 and theorem 2.2, the first on the extend to which the underlying graph of the problem characterizes rigidity and the second on connectivity conditions for rigidity in two dimensions. It is clear however that graph theoretical characterizations do not cover the complete picture, leaving out complete rigidity and also, notably, rigidity in general for three dimensions.

A commonly used characterization for rigidity based on the Jacobian matrix of the measurement function was explained in chapter 4 together with the interpretation of this Jacobian matrix. This characterization is useful for an arbitrary number of dimensions. Also important is the fact that this Jacobian matrix turns out to be central to the structure of both the gradient and Hessian of the cost functions defined. This facilitates analysis of the cost functions.

The derivation of the gradient presented in chapter 3 led to an analogy between the minimization method presented and a system of springs, where the cost function is analogous to the potential energy in the system and the springs are trying to return to their natural length. Using this analogy we argued that the cost functions are not globally convex functions. A simple 1-dimensional example was presented where this is indeed the case. It is easy to extend this example to more complex instances. Some of the instances tested in the computer simulations correspond to these situations. The example also suggests that, at least in some cases, it may be useful to augment the dimensionality of the space in which the frame is defined (space of operations).

Derivation of the Hessian matrix of the cost function revealed a structure that, for an infinitesimally rigid frame, suggested constraints on the independent variables of the function (Grounding, section 4.5) that make the Hessian matrix at the solution points (global minima) positive definite, an advantage in terms of local convergence.

The optimization approach was implemented in software making use of a nonlinear minimization package based on a Model/Trust-Region approach. The method was tested on a few sample networks and the results are presented in the form of graphs (section 4.9). There were instances of convergence and instances in which the algorithm stopped in situations similar to the example refered to above. More experience is also necessary in the use of the software package chosen to achieve more efficient operation.

In chapter 5 we considered the effect of small errors in the distance measurement information available. This was done by considering the effect of these errors on the loci of the global minima of the cost functions. By small we have meant that a first order approximation to the gradient of the cost function using Taylor's theorem is valid. The gradient can then be expressed, locally, as a linear function of the measurements errors, the vector d, and the errors induced on the final positions determined for the nodes of the network, the vector q. (section 5.3.) We are able then to derive a linear relationship between d and q. The dependence of this relationship on the choice of cost function and on the final positions of the nodes is apparent from this expression.

For special choices of cost function (section 5.4) the relationship derived has an intuitive explanation, namely that the new local minima correspond to points (solutions) that best achieve an approximation, for the given Grounding, to the changes in edge lengths (of the underlying graph) introduced by the measurement errors. The way Grounding is implemented however influences the dependence of the method on the measurement errors (section 5.5).

6.3 FURTHER RESEARCH

More research would be needed to compare the complexity of IPLS to that of an optimization approach for cases where IPLS is capable of finding the solution.

Also the question of whether the optimization approach can be made to converge to a valid solution in general is still open. As mentioned, augmenting the dimensionality of the space of operation can be a useful approach. Since we have expressions for the gradient and Hessian matrices of the cost functions at all points it may be possible to investigate in general the effect of an augmented space of operations on stationary points.

Use of an augmented space of operations introduces the problem of having to guarantee that the iterations in the minimization process make the frame under iteration fold back onto a space with the initial number of dimensions of the network (the space of solutions). Modifying the cost function in this instances so that position coordinates outside the space of solutions carry some appropriate cost may be a way around this problem.

Since the emphasis of the research was on analysis of the properties of the cost function topics like rate of convergence of the minimization method where not treated here. The number of iterations required by the software we developed is large and research should be done on how to optimize the software package to achieve faster convergence. Similarly, research can be done on the effect of the structure of the Hessian and gradient of the function on the rate of convergence of the minimization method chosen. More research would also be required to investigate the rate of convergence as a function of the number of nodes for arbitrary networks.

At the moment grounding of the frame in the algorithm is done to some extend in an arbitrary fashion. Research on how to choose the best form of grounding may give useful results since the choice of grounding may affect the speed of convergence and also the sensitivity of the algorithm to measurement errors. Also research on choosing the initial frame from which the algorithm starts to iterate may lead to a way of choosing an initial frame more likely to be close to the solution and hence improve the typical number of iterations required for convergence of a given instance of the PL problem.

We have not discussed here network implementations of a PL system. For example a distributed algorithm for Postion Location in a large network using locally the optimization approach is possible, just as for IPLS. Even if a local subset of nodes fed into the algorithm corresponds to a Loose frame the final realization arrived at may serve as a good initial realization to be used together with the positions computed for a neighboring subset of nodes. Use of a distributed algorithm in a large network would have the effect of relieving nodes processing the information of a great deal of computation. Besides, in a large network, most of the measurements available in a dense subset of nodes are likely to be between nodes within this subset. Hence not much critical information, as far as the subset is concerned, would be expected from measurements related to nodes outside this subset. Distributed processing by several subsets of this kind (dense) may be the most appropriate way to implement a PL algorithm in the large network.

Once a better understanding of the speed of convergence of the iterative approach to PL problems is obtained it would also be useful to make calculations to determine how well such a system would perform in real time for networks with mobile Packet Radio Units, specially since this is one of the most attractive features of a Packet Radio Network.

APPENDIX

Expressions for the individual entries in the Hessian matrix of the cost functions, types A and B

Designate by H(Q) the matrix of second order partial derivatives with respect to the free variables in C(Q). (The Hessian matrix of C(Q).) We use the same notation as in the previous chapters.

$C^A(Q)$: Type A cost function. Similarly for type B.

r_{ij}: The "edge vector" corresponding to edge (i,j), obtained from the realization by $r_{ij} = r_i - r_j$.

u_{ij}: The unit vector pointing in the direction r_{ij}.

e_{ij}: The "edge error" corresponding to edge (i,j). It is particular to each type of cost function.

A_{ij}: The "edge weight" corresponding to edge (i,j) for type A cost function ($A_{ij}>0$). Similarly for type B.

p_A: A normalization constant for type A cost function ($p_A>0$). Similarly for type B.

Elements of the Hessian matrix are denoted as follows.

$$H_{ixjy} = H_{ixjy}(Q) \equiv d/djy\{\ d/dix\{C(Q)\}\ \} = H_{jyix}$$

$$= d/djy\{\ g_{ix}(Q)\ \}$$

Five different situations are possible when obtaining expressions for the elements H_{ixjy} of the matrix H(Q).

case 1: $i=j$, $x=y$

case 2: $i=j$, $x \neq y$

case 3: j element of Ni, $x=y$

case 4: j element of Ni, $x \neq y$

case 5: $j \neq i$, j not in Ni.

Below we show the derivation for the individual entries for the Hessian matrix of the cost functions defined in Chapter 3. The resulting equations appear in Table 4.1 of chapter 4.

Type A cost function.

Equations 3.4.a, 3.5.a and 3.6.a contain the expressions for the gradient $g^A(Q)$.

<u>case 1</u>: $j=i$, $x=y$

$$H^A_{ixjy} = H^A_{ixix} = d^2/(dix)^2\{ C^A(Q) \}$$
$$= (4/p_A) \text{ SUM}_{k \text{ in } Ni}[A_{ik}(e_{ik} \; d/dix\{r_{ikx}\} + r_{ikx} d/dix\{e_{ik}\})]$$
$$= (4/p_A) \text{ SUM}_{k \text{ in } Ni}[A_{ik}(e_{ik} \; (1) \quad + \quad 2 \; r_{ikx} \; r_{ikx})]$$
$$= (4/p_A) \text{ SUM}_{k \text{ in } Ni}[A_{ik}(e_{ik} + 2(r_{ikx})^2)] \qquad 4.9.a$$

<u>case 2</u>: $j=i$, $x \neq y$

$$H^A_{ixjy} = H^A_{ixiy} = H^A_{iyix} = d^2/_{dixdiy}\{ C^A(Q) \}$$
$$= (4/p_A)\text{SUM}_{k \text{ in } Ni}[A_{ik}(e_{ik} \; d/dix\{r_{ikx}\} + r_{ikx} \; d/diy\{e_{ik}\})]$$
$$= (4/p_A) \text{ SUM}_{k \text{ in } Ni}[A_{ik}(\quad 0 \quad + \quad 2 \; r_{ikx} \; (r_{iky}))]$$
$$= (8/p_A) \text{ SUM}_{k \text{ in } Ni}[A_{ik}(r_{ikx} r_{iky})] \qquad 4.10.a$$

<u>case 3</u>: j in Ni, $x=y$

$$H^A_{ixjy} = H^A_{jyix} = d^2/_{dixdjx}\{ C^A(Q) \}$$

$$= (4/p_A)\mathrm{SUM}_{k \text{ in } Ni}[A_{ik}(\ e_{ik}\ d/djx\{r_{ikx}\} + d/djx\{e_{ik}\}\ r_{ikx}\ )]$$

when k ≠ j both derivatives in the expression are zero, hence

$$H^A_{ixjx} = (4/p_A)\ [\ A_{ij}(\ e_{ij}\ d/djx\{r_{ijx}\} + d/djx\{e_{ij}\}\ r_{ijx}\ )\ ]$$

$$= (4/p_A)\ [\ A_{ij}(\ e_{ij}(-1) + (-2\ r_{ijx})\ r_{ijx}\ )\ ]$$

$$= -(4/p_A)\ [\ A_{ij}(\ e_{ij} + 2\ r_{ijx}^{\ 2}\ )\ ] \qquad 4.11.a$$

<u>case 4</u>: j in Ni, x≠y

$$H^A_{ixjy} = H^A_{jyix} = d^2/_{dixdjy}\{\ C^A(Q)\ \}$$

$$= (4/p_A)\ \mathrm{SUM}_{k \text{ in } Ni}[A_{ik}(\ e_{ik}\ d/djy\{r_{ikx}\} + d/djy\{e_{ik}\}\ r_{ikx}\ )]$$

$$= (4/p_A)\ [\ A_{ij}(\ e_{ij}\ d/djy\{r_{ijx}\} + d/djy\{e_{ij}\}\ r_{ijx}\ )\ ]$$

$$= (4/p_A)\ [\ A_{ij}(\ e_{ij}\ \ (0)\ + (-2\ r_{ijy})\ \ r_{ijx}\ )\ ]$$

$$= -(8/p_A)\ A_{ij}\ r_{ijx}r_{ijy} \qquad 4.12.a$$

<u>case 5</u>: j≠i , j not in Ni

$$H^A_{ixjy} = 0 \qquad \text{(by inspection)} \qquad 4.13.a$$

Type B cost function.

Equations 3.4.b, 3.5.b and 3.6.b contain the expressions for $g^B(Q)$.

<u>case 1</u>: j=i , x=y

$$H^B_{ixjy} = H^B_{ixix} = d^2/_{(dix)^2}\{\ C^B(Q)\ \}$$

$$= (2/p_B)\ \mathrm{SUM}_{k \text{ in } Ni}[\ B_{ik}\{\ e_{ik}\ (||r_{ik}|| - r_{ikx}u_{ikx})/||r_{ik}||^2$$

$$+ \quad r_{ikx}r_{ikx}/||r_{ik}||^2\ \}\ ]$$

$$= (2/p_B)\mathrm{SUM}_{Ni}[B_{ik}\{(u_{ikx})^2 + (1-u_{ikx}^{\ 2})e_{ik}/||r_{ik}||\}] \qquad 4.9.b$$

case 2: $j=i$, $x \neq y$

$$H^{B}{}_{ixjy} = H^{B}{}_{ixiy} = H^{B}{}_{iyix} = d^{2}/_{dixdiy}\{ C^{B}(Q) \}$$

$$= (2/p_{B})\ \mathrm{SUM}_{k \text{ in } Ni}[\ B_{ik}(\ 0 \ + \ u_{ikx}u_{iky}\ )\ ]$$

$$= (2/p_{B})\ \mathrm{SUM}_{k \text{ in } Ni}[\ B_{ik}\ u_{ikx}u_{iky}\ ] \qquad 4.10.b$$

case 3: j in Ni, $x=y$

$$H^{B}{}_{ixjy} = H^{B}{}_{jyix} = d^{2}/_{dixdjx}\{ C^{B}(Q) \}$$

$$= -(2/p_{B})\ [B_{ij}\{(u_{ijx})^{2} + (1-u_{ijx}{}^{2})\ e_{ij}/||r_{ij}||\}] \qquad 4.11.b$$

case 4: j in Ni, $x \neq y$

$$H^{B}{}_{ixjy} = H^{B}{}_{jyix} = d^{2}/dixdjy\{ C^{B}(Q) \}$$

$$= -(2/p_{B})\ [\ B_{ij}\ u_{ijx}u_{ijy}\ ] \qquad 4.12.b$$

case 5: $j \neq i$, j not in Ni

$$H^{B}{}_{ixjy} = 0 \qquad \text{(by inspection)} \qquad 4.13.b$$

In the matrix H(Q) the ordering of the variables is taken to be consistent with that of g(Q). We can now use this equations to arrive at equation 4.14.

Let H_{ij} be the m by m submatrix of H(Q) having entries H_{ixjy}. Let I denote the identity matrix. Let U_{ik} be a column matrix of vector entries $\underline{u}_{ikx}$, such that
$\underline{u}_{ikx}{}^{t}\underline{u}_{iky} = u_{ikx}{}^{2}$ for y=x and zero for y≠x. We can then write:

For j=i:

$$H^{A}ij= H^{A}ii = (4/p_{A})\mathrm{SUM}_{k \text{ in } Ni}[2A_{ik}r_{ik}r_{ik}{}^{t} + A_{ik}e_{ik}\ I] \qquad \text{A.1.a}$$

$$H^{B}{}_{ij}= H^{B}{}_{ii} = (2/p_{B})\mathrm{SUM}_{k \text{ in } Ni}[\ B_{ij}\ u_{ik}\ u_{ik}{}^{t} + (B_{ik}e_{ik}/||r_{ik}||)(I - U_{ik}U_{ik}{}^{t})\ ] \qquad \text{A.1.b}$$

For j element of Ni

$$H^{A}{}_{ij}= -(4/p_{A})[\ 2\ A_{ij}\ r_{ij}\ r_{ij}{}^{t} + A_{ij}\ e_{ij}\ I\ ] \qquad \text{A.2.a}$$

$$H^{B}{}_{ij}= -(2/p_{B})[\ B_{ij}\ u_{ij}\ u_{ij}{}^{t} + (B_{ij}e_{ij}/||r_{ij}||)\ (I - U_{ij}U_{ij}{}^{t})\ ] \qquad \text{A.2.b}$$

For j≠i, j not in Ni

$$H^{A}{}_{ij}= \text{zero-matrix} \qquad \text{A.3.a}$$

$$H^{B}{}_{ij}= \text{zero-matrix} \qquad \text{A.3.b}$$

From equations A.1 and A.2 we can see that all non-zero submatrices H_{ij} of H(Q) (for types A or B) are the sum of two matrices, one which depends solely on the position vectors of the vertices and the other which also depends on the edge errors e_{ij}. We note that the portion dependent on the edge errors contains the terms e_{ij} as factors of all its elements and thus becomes zero when its edge error is zero.

We can write

$$H^A{}_{ij} = (4/p_A)\ (2\ P^A{}_{ij} + E^A{}_{ij})$$
$$H^B{}_{ij} = (2/p_B)\ (P^B{}_{ij} + E^B{}_{ij})$$

where the matrices P_{ij} are dependent only on the position vectors of the vertices (not to be confused with the notation for Positions of a frame) and E_{ij} are the matrices having the edge error as a factor. Similarly we can write

$$H^A(Q) = (4/p_A)\ (2\ P^A + E^A) \qquad 4.14.a$$
$$H^B(Q) = (2/p_B)\ (P^B + E^B) \qquad 4.14.b$$

where P and E are the augmented versions of P_{ij} and E_{ij}.

REFERENCES

[AR] L.ASIMOW and B.ROTH,"The Rigidity of Graphs,I,II,"Preprint, University of Wyoming,Laramie,1977-78.

[B] B.BOLLOBAS,"Graph Theory an introductory course," Springer-Verlag New York Inc.,New York,1979.

[BR] O.BOTTEMA and B.ROTH,"Theoretical Kinematics,"North-Holland Publishing Company,New York,1979.

[C] C.T.CHEN,"Introduction to Linear System Theory,"Holt,Rinehart and Winston, Inc.,New York,1970.

[DS] J.DENNIS and R.SCHNABEL,"Numerical Methods for Unconstrained Optimization and Nonlinear Equations," Prentice-Hall,Inc.,New Jersey,1983.

[G 80] D.GAY,"Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach," Technical Report #TR-18, Sloan School of Management,MIT, Oct.1980.

[G 83] D.GAY,"Algorithm 611. Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach," *ACM Transactions on Mathematical Software*, Vol.9,No.4, pp.503-524, Dec.1983.

[Gl] H.GLUCK,"Almost All Simply Connected Closed Surfaces are Rigid," Geometric Topology (Lecture Notes in Mathematics, 438), Springer-Verlag,Berlin, 1975,pp.225-239.

[GMR] GREENBERG,S.MELVIN and J.ROME,"Stability Analysis of Relative Navigation Systems," *Position Location and Navigation Symposium*, pp.335-344, IEEE,1978.

[H] P.HALMOS,"Finite-Dimensional Vector Spaces," 2nd printing, Springer-Verlag New York Inc.,New York,1974,pp.145,142.

[Ha] F.HARARY,"Graph Theory," 3rd printing,Addison-Wesley Publishing Company,Mass.,Oct 1972.

[He] T.HENRY,"Acoustic Transponder Navigation," *Position Location and Navigation Symposium*, pp.237-244, IEEE,1978.

[K et al] R.KAHN et al.,"Advances in Packet Radio Technology," *Proceedings of the IEEE 66* (11), November 1978,pp.1468-1496.

[KMM] J.KIVETT,F.MORSE and R.MONZINGO,"Implementation of Tracking Filters in the PLRS Master Unit," *Position Location and Navigation Symposium*, pp.283-297, IEEE,1978.

[KS] B.KRIEGSMAN and W.STONESTREET,"A Navigation Filter for an Integrated GPS/JTIDS/INS System for a Tactical Aircraft," Position Location and Navigation Symposium, pp.237-244, IEEE,1978.

[L] G.LAMAN,"On Graphs and Rigidity of Plane Skeletal Structures," Journal of Engineering Mathematics 4 (1970),331-340.

[LY] L.LOVASZ and Y.YEMINI,"On Generic Rigidity in the Plane," SIAM Journal of Algebraic and Discrete Methods,Vol 3,No.1,March,1982,91-98.

[N] L.NEWMAN,"The Development of JTIDS Relative Navigation from advanced development to full operational potential," Position Location and Navigation Symposium, pp.305-309, IEEE,1978.

[R] R.ROBERTS,"The Development of PLRS User Equipment," Position Location and Navigation Symposium, pp.276-282,IEEE,1978.

[S] G.STRANG,"Linear Algebra and its applications," 2nd edition, Academic Press,Inc., New York,1980.

[SS] W.STEELE and J.SCHLENGER,"JTIDS Provides a Relative Navigation/Communications Approach to Multi-Sensor Integration," Position Location and Navigation Symposium, pp.310-315,IEEE,1978.

[T] A.TANENBAUM,"Computer Networks," Prentice-Hall,Inc., New Jersey,1981,p.53.

[Y 79] Y.YEMINI,"Some Theoretical Aspects of Position-Location Problems,"20th Annual Symposium on Foundations of Computer Science, pp.1-8,IEEE, San Juan, PR, Oct.1979.

[Y 81] Y.YEMINI,"IPLS-An Incremental Position Location Algorithm," Draft,California,Jan.1981.