MODELING AND ANALYZING THE UNCERTAINTY PROPAGATION IN VECTOR-BASED NETWORK STRUCTURES IN GIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OYA YARKINOĞLU GÜCÜK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
GEODETIC AND GEOGRAPHIC INFORMATION TECHNOLOGIES

SEPTEMBER 2007

Approval of the thesis:

MODELING AND ANALYZING THE UNCERTAINTY PROPAGATION IN VECTOR-BASED NETWORK STRUCTURES IN GIS

submitted by **OYA YARKINOĞLU GÜCÜK** in partial fulfillment of the requirements for the degree of **Master of Science in Geodetic and Geographic Information Technologies Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Assoc. Prof. H. Şebnem Düzgün _____
Head of Department, **Geodetic and Geographic Information Technologies**

Assoc. Prof. H. Şebnem Düzgün
Supervisor **Geodetic and Geographic Information Technologies Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Nurünnisa Usul _____
Civil Engineering Dept., METU

Assoc. Prof. Dr. H. Şebnem Düzgün _____
Mining Engineering Dept.., METU

Assist. Prof. Dr. Zuhal Akyürek _____
Civil Engineering Dept., METU

Assoc. Prof. Dr. Mahmut Onur Karslıoğlu _____
Civil Engineering Dept., METU

Assist. Prof. Dr. Ayşegül Aksoy _____
Environmental Engineering Dept.., METU

Date: (_____)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Oya YARKINOĞLU GÜCÜK

Signature :

ABSTRACT

MODELING AND ANALYZING THE UNCERTAINTY PROPAGATION IN VECTOR-BASED NETWORK STRUCTURES IN GIS

GÜCÜK, Oya YARKINOĞLU

M.S., Department of Geodetic and Geographic Information Technologies

Supervisor : Assoc. Prof. Dr. H. Şebnem Düzgün

September 2007, 131 pages

Uncertainty is a quantitative attribute that represents the difference between reality and representation of reality. Uncertainty analysis and error propagation modeling reveals the propagation of input error through output.

Main objective of this thesis is to model the uncertainty and its propagation for dependent line segments considering positional correlation.

The model is implemented as a plug-in, called Propagated Band Model (PBM) Plug-in, to a commercial desktop application, GeoKIT Explorer. Implementation of the model is divided into two parts. In the first one, model is applied to each line segment of the selected network, separately. In the second one, error in each segment is transmitted through the line segments from the start node to the end node of the network. Outcomes are then compared with the results of the G-Band model which is the latest uncertainty model for vector features. To comment on similarities and differences of the outcomes, implementation is handled for two different cases. In the first case, users digitize the selected road network. In the second case recently developed software called Interactive Drawer (ID) is used to allow user to define a new network and simulate this network through Monte Carlo Simulation Method. PBM Plug-in is designed to accept the outputs of these implementation cases as an input, as well as generating and visualizing the uncertainty bands of the given line network.

Developed implementations and functionality are basically for expressing the importance and effectiveness of uncertainty handling in vector based geometric features, especially for line segments which construct a network.

Keywords: Vector-Based Uncertainty, Uncertainty Propagation, Geographic Information Systems, Covariance Based Error Band, G-Band, Line Network

ÖZ

CBS'DE VEKTÖR TABANLI AĞ YAPILARINDA BELİRSİZLİĞİN YAYILIMININ MODELLENMESİ VE ANALİZİ

GÜCÜK, Oya YARKINOĞLU

Yüksek Lisans, Jeodezi ve Coğrafi Bilgi Teknolojileri Bölümü

Tez Yöneticisi : Doç. Dr. H. Şebnem DÜZGÜN

Eylül 2007, 131 sayfa

Belirsizlik nicel bir özellik olup, gerçeklik ve onun gösterimi arasındaki farktır. Belirsizlik analizi ya da hata yayılımı modellemesi girdi hatasının çıktılara etkisini göstermektedir.

Tezin temel amacı belirsizlik ve onun yayılımını konumsal korelasyonu dikkate alarak bağımlı çizgi parçaları için modellemektir.

Modelin uygulaması Propagated Band Model (PBM) Eklentisi adıyla, ticari bir masaüstü uygulaması olan GeoKIT Explorer'a eklenti olarak yazılmıştır. Modelin uygulaması iki parçaya ayrılmıştır. İlkinde, model seçilen çizgi ağını oluşturan her çizgi parçası için ayrı ayrı uygulanmıştır. İkinci kısımda ise aktarılan hata, çizgi parçaları boyunca iletilerek başlangıç noktasındaki hatanın son noktaya değin yayılımı modellenmiştir. Sonuçlar vektör nesneleri için en son önerilen G-Band modelin sonuçları ile karşılaştırılmıştır. Sunulan sonuçların benzerlik ve farklılıkları üzerine yorum yapabilmek için bazı testler yapılmıştır. Bu testler iki farklı şekilde ele alınmıştır. İlk kısımda seçilmiş olan yol ağı kullanıcılarca sayısallaştırılmıştır. İkinci kısımda ise yeni geliştirilmiş bir yazılım olan Interactive Drawer (ID) kullanılarak kullanıcının kendi çizgi ağını tanımlamasına ve Monte Carlo Simulasyonu kullanarak benzetişiminin yapılmasına olanak sağlanmıştır. PBM eklentisi diğer uygulamaların çıktılarını girdi olarak kabul edebilecek şekilde tasarlanmıştır. Buna ek olarak geliştirilen yazılım

verilen çizgi ağı için belirsizlik bantlarını oluşturabilmekte ve görüntüleyebilmektedir.

Geliştirilen uygulama, vektör tabanlı geometrik objelerdeki ve çizgi ağlarındaki belirsizliğin ele alınmasının önemini ve etkilerini vurgulamak amaçlıdır.

Anahtar Kelimeler: Vektör-Tabanlı Belirsizlik, Belirsizlik Yayılımı, Coğrafi Bilgi Sistemleri, Kovaryans (Eşdeğişirlik) Tabanlı Hata Bandı, G-Bandı, Çizgi Ağı

To my family

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Şebnem DÜZGÜN, for her support, collaboration, and guidance. I also would like to thank Bilgi GIS family for the technical support they provided and their sensibility.

I sincerely thank to the people who digitize the line network which I used in my thesis. I also would like to thank Ahmet Kursat KURTAR, Murat Durmaz, Özge Ergin, and GGIT staff. Without their help, it would be difficult to overcome the problems faced throughout my MSc. study and thesis research.

Special thanks are given to my parents for their support and encouragement. Finally, I would like to thank my husband Osman GÜCÜK for his moral support and patiance. His sensibility undoubtedly gave me extra strength to finalize this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Geographic Information System (GIS) is a computer-based system which is capable of storing, manipulating, displaying geographic information handled from abstraction of real world entities. As it is an abstraction, it is difficult to introduce geographical data to GIS without any distortions and errors.

Importance of GIS arises from the need to manage huge amount of spatial data. Furthermore revising, searching, updating, analyzing and sharing data are less problematic with the use of GIS. But this time the quality of spatial data stored in the GIS databases is questioned. Therefore this is the starting point of error and uncertainty researches.

In the following sections, definition of error and uncertainty, sources of error in GIS, and developed models to manage error and uncertainty are reviewed.

1.1 Measurement Error (ME) and Uncertainty

In this section, error and uncertainty concepts are clearly defined and relationship between error and uncertainty is explained. Also types and sources of error and uncertainty are stated.

1.1.1 Measurement Error

Rabinovich (2000) defines *measurement* as a process of finding a numerical representation of a physical quantity with the help of *measuring instruments*. Measurements are mainly categorized as direct and indirect measurements. In direct measurements, object of interest has strong relations with the measuring device and read value from the instrument is directly assigned to the object. However, in indirect measurements

quantity of the object is extracted from the mathematical relation that exists between the object and its arguments. For example, speed of a car can be calculated by dividing the measured travelled distance with measured travel time.

As measuring devices are human made objects, accuracy of the results obtained from these devices has to be discussed. Thus, imperfection which is introduced during the measurement process can be characterized quantitatively by using the limits of error and uncertainty. Deviation of measurement results from the true value of the measurable quantity is defined as *measurement error* and is categorized as either relative or absolute (Rabinovich, 2000).

Absolute error is a physical quantity which may be positive, negative or given by an interval. It has the same units of the measurable quantity and expressed as the difference between measurement result and true value of the measurable object. If A is the true value of the measurable quantity and $\widetilde{A}$ is the measurement result, then absolute error of measurements can be formulated as;

$$\varsigma = \widetilde{A} - A \tag{1.1}$$

The relative error is expressed as a fraction of true value of the measurable quantity. Relative errors are represented as percent and generally small errors existing in the precise measurements are expressed as relative error. Relative error is formulated as;

$$\epsilon = \frac{\widetilde{A} - A}{A} \tag{1.2}$$

where A is the true value of the measurable quantity and $\widetilde{A}$ is the measurement result and ϵ is the relative error.

As mentioned before, the measurement error is deviation from the mean value of the sample which approximates the mean value of the population. Although the absolute measurement error given in equation 1.1 focuses on the difference between measured and best estimate of the true mean, usually this equation could not find an implementation in GIS because the true value of the measured quantity is usually unknown for many cases. In this case, instead of the true value, a measurable quantity with sufficient accuracy can be used to estimate the measurement error from the data set (Rabinovich, 2000).

Rabinovich (2000) categorizes the necessary components of measurement and explains the related measurement errors as;

1. *Measurement method* which introduces the methodological error, ς_m

2. *Measuring instrument* which introduces the instrumental error, ς_i

3. *Measurement participator* or person who introduces the personal error, ς_p

Finally, the general formula for the measurement error, ς is;

$$\varsigma = \varsigma_m + \varsigma_i + \varsigma_p \tag{1.3}$$

Main reason for methodological error is generally caused from misinterpretation of the relations which are employed to find an estimate of the measurable quantity. Rise of methodological errors are often related to incompleteness or insufficiency of the theory on which the measurement is based. The instrumental measurement errors are caused by the imperfection of measuring instrument. Normally the intrinsic error of measuring instruments under reference conditions are regarded as normal and they are distinguished from additional error. Another error source is the human factor which is mostly introduced in GIS during data collection and manipulation process. Measurements are often performed by people. So individual characteristic of the person performing the measurement give rise to individual errors. Sometimes incorrect reading, sometimes inattention or negligence of the person could be reasons for personal error (Rabinovich, 2000).

Another classification can be done for measurement errors considering the cause of error. A measurement error is categorized as *systematic*, if it remains constant or changes regularly in repeated measurements. The observed and estimated systematic error is eliminated from measurement by introducing corrections. If some quantity is measured several times and incoming differences on the results of these separate measurements could not be predicted or could not show a regular fashion then these scattered results are called *random errors* (Rabinovich, 2000).

Random errors are discovered by performing repeated measurement of the same quantity under the same conditions. On the other hand systematic errors can be discovered experimentally either by comparing the measurement results performed

by the use of a different method or by using a more calibrated measuring instrument. Although, systematic errors can be estimated by different methods, based on the known properties of the measurand and the measuring instruments, estimation of random errors are based on obtained data. Random error estimation is done indirectly by using standard deviation which characterizes the distribution of the random error within the data (Rabinovich, 2000).

Errors in GIS are generally grouped as; positional, attribute, topological and temporal errors. Burrough and McDonnel (1997) categorize the sources of GIS errors as:

- Obvious sources of error; age of data, map scale, density of observation, format, relevance, areal cover, accessibility and cost.
- Errors resulting from natural variations or from original measurements; positional accuracy, accuracy of content and sources of variation in data.
- Errors arising through processing; numerical errors, errors in topological analysis, classification and generalization problems and digitizing and geocoding errors.

1.1.2 Uncertainty

The positional accuracy of a spatial object can be evaluated by getting the difference between the measured location and the most accurate location (Bley and Haller, 2006). However identifying the most accurate locations are difficult and problematic because of "measuring instruments, frames of geodetic reference and feature definitions", "the positional accuracy of a feature practically is defined through measures of the difference between the location recorded in the database and a location determined with a higher accuracy" (Goodchild and Hunter, 1997).

Zhang and Goodchild (2002) unify uncertainty with ambiguity, inexactness and vagueness that exist in geographical data which reflects the complexity of the real world. Uncertainty may be introduced by the absence of information and amount of uncertainty is related to the information needed to reach the truth.

Ayyub and Chao (1997) classify uncertainty as either objective or subjective. *Ambiguity* lies in objective uncertainty type. Physical, statistical and modeling uncertainty are sources of ambiguity. Subjective type of uncertainty or *vagueness* includes

expert-based assessment, human error, etc. The ambiguity in objective uncertainty are generally due to non-cognitive reasons like:

1. Physical randomness;

2. Statistical uncertainty caused from the use of sampled information to estimate the model response characteristics;

3. Lack of knowledge; and

4. Modeling uncertainty that is due to simplifying assumptions of real performances

In the presence of ambiguity, probability can be used to model non-cognitive type of uncertainty. For vagueness-related uncertainty the situation is different. Because vagueness-related uncertainty is due to cognitive sources (Ayyub and Chao, 1997) like the definition of certain parameters, human factors, and defining the interrelationships among parameters for complex systems. Also analyzing the probability is not easy for vagueness-related uncertainty. For this reason, during the analysis of vagueness either fuzzy approaches or both fuzzy-random approaches should be preferred (Ayyub and Chao, 1997).

Klir and Yuan (1995) also categorize the uncertainty into fuzziness and ambiguity, considering the imperfection and incompleteness of the available data. They also divide ambiguity into two: conflict and nonspecificity. Measurements from observations of repeated experiments are often precise but are in conflict because conflict causes results of different outcomes. Nonspecificity occurs when multiple outcomes of an action are left unspecified. Fisher (1999) defines uncertainty which is dependent upon how well objects or classes are defined.

In the preceding paragraphs, classification of uncertainty is given in a generalized way, due to sources of cause and missing or deficient information. However, rather than classifying the uncertainty, analyzing and managing the uncertainty involved in the system is important. In the following part of this chapter uncertainty analysis in GIS and GIS related subjects are explained.

1.1.3 Relationship Between Error and Uncertainty

A variety of terms are used to define uncertainty. Some of them are ; error, accuracy, precision, vagueness, ambiguity, and reliability. Accuracy refers to an agreement between measured and the true value. In precision, it is not required to know the true value because it refers to the repeatability of measurement. According to Rabinovich (2000) error and uncertainty are used with the same meaning for a long period of time but these concepts need to be clarified.

Error is disagreement between a measurement value and the true or accepted value. True value of the measurable object has to be known to discuss error. But in science it is difficult to discuss error because in most of the cases true value is not known. Uncertainty of a measured value is an interval around this value such that any repetition of the measurement will produce a new result that lies within this interval. The uncertainty interval is decided by the experimenter through consideration of the uncertainty estimation principles (Bellevue_Community_College, 2007).

Uncertainty, allows the scientist to make completely certain statements. Stating a confidence interval for a measurement allow the scientist to make statements. Additionally, uncertainties may also be stated along with a probability. For this case the measured value has the stated probability to lie within the confidence interval (Bellevue_Community_College, 2007).

For example if the measurement result is different from the true value of the measured quantity, for the case at which the true value is known according to the calculated difference, error can be expressed as +1%, but for the case of unknown true value, the calculation can be handled by using the most accurate value decided after a repeated measurement process and express error as $\pm 1\%$ (Rabinovich, 2000). In this case instead of the word *error*, the word *uncertainty* has to be used. This plus minus representation means that measurement error is simultaneously both +1% and -1%. The format is called "value plus or minus uncertainty" (Bellevue_Community_College, 2007).

1.2 Modeling Uncertainty and Its Propagation

Choi (2005) defines uncertainty analysis as estimating output uncertainty considering the variability in input parameters. Bonin (2006) defines uncertainty analysis as

a general study of anaylzing the uncertainty in the output of spatial models with having a certain knowledge about input uncertainty analysis and sensitivity analysis (Fisher, 1991; Heuvelink, 1993; Heuvelink and Burrough, 1993; Veregin, 1992). According to Lilburne et al. (2006) uncertainty analysis can be applied to uncertainty caused by imprecise environmental process and input data based model predictions. However sensitivity analysis tries to determine which input uncertainty is important in the determination of output uncertainty.

Depending on the types of uncertainty analysis methods, Choi (2005) classifies the types as non-probabilistic and probabilistic methods. Non-probabilistic methods include Interval Analysis and Fuzzy Logic methods, which are useful for a deterministic system model. In deterministic system model, there are no random errors in response of the system and input parameters are uncertain without probability density functions which represents the lack of information. If a system model is deterministic, input parameters are known as probability density functions. If a system model is non-deterministic and input parameters are known as probability density functions, then probabilistic methods for uncertainty analysis are also useful for uncertainty quantification.

1.3 Problem Definition

In GIS, geographic information can be represented either in raster model or in vector model. Uncertainty in raster model can raise from poor class definition, mixed pixels and transition zones. To model and quantify uncertainty in raster GIS, fuzzy and probabilistic techniques can be helpful.

Fuzzy sets are defined by Zadeh (1965) and form the bases of fuzzy classification. Wang (1990), Foody (1996) and Zhang and Foody (2001) proposed fuzzy classification approaches to adapt to the natural phenomena. In recent years rather than applying fuzzy classification, some visualization prototypes have been proposed by Wel et al. (1997), Blenkinsop et al. (2000), Bastin et al. (2002), and Lucieer and Kraak (2003) to present and explore uncertainty in remotely sensed image classification.

Buffer analysis is one of the application area on which studies of error propagation is carried. Veregin (1992) derived a thematic error propagation model for raster-based GIS using proportion of correctly classified pixels (PCC) which were used as a

measure of thematic error. Veregin (1996) extended the model to a probability-based raster database to handle class specific thematic errors. Genst and Canters (2000) studied error propagation in a raster environment by Monte Carlo simulation with the aid of a fuzzy membership function. In this research to reflect the classification uncertainty, class membership properties were estimated. Additionally, Zhang et al. (1998) derived an error propagation model of a buffer in a vector environment based on E-band models, which were used for describing the positional error of spatial features in GIS.

As the subject of this thesis is to model uncertainty and its propagation for vector-based GIS objects which build up a network feature, following paragraphs review previously proposed uncertainty models for vector-based GIS. In vector-based GIS, spatial features are represented by point, line and polygon. As point is the primitive element of geometric objects, points and their error models have been studied in the field very often. Consequently, line and polygon uncertainty models are based on the point uncertainty models. In the literature several different uncertainty models exist which are mainly for point based uncertainty modeling (Caspary and Scheuring, 1992; Kurtar, 2006; Perkal, 1956, 1966; Shi, 1994; Shi and Liu, 2000; Wolf and Ghilani, 1997).

Perkal (1956, 1966) proposed an error band model which provides a boundary surrounding the line segment. It is proposed that true line lies within the band with a propabability of 1.0 and never deviates outside. Chrisman (1982) extends the work of Perkal (1966) and define epsilon band which is based on a constant radius of epsilon around the line. Epsilon is calculated from the radius of the line's endpoint error circles.

Caspary and Scheuring (1992) proposed an error band which become narrower at the midpoint of the line segment. Dutton (1992) uses Monte Carlo simulation and error propagation law to derive the band. Error at the end points of the line are assumed to be independent and equal. Two years later, Shi (1994) defined a new model under the assumption of coordinate errors of uncorrelated end points which are following a normal distribution. In this model, error distribution of the end points effect the positional error of the selected point along the line. Shi (1994) define two end points as stochastic vectors following a normal distribution, these vectors are four dimensional vectors of equal variances and covariances.

Wolf and Ghilani (1997) proposed error ellipses which use standard deviations in x and y directions of the point to define semi-major and semi-minor axes. Direction of the error ellipse is determined due to the dependency between coordinate pairs x and y. Error ellipses visually represent the uncertainty region of a point. This is the idea of today's band models which use the uncertainty of each point along the line to construct a region of uncertainty surrounding the entire line.

Shi and Liu (2000) developed the idea of Shi (1994) and defined a model called G-Band Model which considers the correlation between end points during the creation of the band. They assume that the condition is different at end point errors and derive the errors of arbitrary points on the line to find out a distribution and density function for the whole line.

Kurtar (2006) extends G-Band model and models the uncertainty of functional curves which are generated by non-linear functions. In his study, proposed uncertainty bands for arc by three coordinates, arc string and cubic spline are based on G-Band model approach. Kurtar (2006) proposes epsilon band model approach for arc by center point and clothoid because these curves are created with a single coordinate together with other scalar variables.

Up to here, the defined band models for vector-based GIS, model how the uncertainty at the end points of the line segment propagates along the line's interior. Model proposed by Leung et al. (2004) is a covariance-based error band model which derives the uncertainty of point from the covariance matrix associated with it and error ellipses are used to represent the uncertainty of the point. However, the originality introduced in this model is that it investigates the error propagation through the line from the start point to the end point of the line segment.

All these models for vector-based GIS are implemented only for a single line segment. There is not any model to estimate the error propagation of a network, which is made up of combination of line segments. Within the study of this thesis, a new method is developed to model the uncertainty propagation in a network feature which is composed of line segments. Uncertainty model proposed by Leung et al. (2004) is combined with this method to visualize the uncertainty which propagates from the start node to the end node of the network.

The main objective of this thesis is to model uncertainty propagation in a network feature which is constructed from line segments for this purpose, a new method is

proposed to model uncertainty propagation in network feature. In this method, uncertainty propagation idea introduced in the study of Leung et al. (2004) is used. According to Leung et al. (2004), to model uncertainty propagation in a line segment, end node of the segment has to be defined as a function of start node.

In this method for the line segment given in Figure 1.1 x and y components of the coordinate at the end point of the segment displayed by $V(x)$ is derived by using Equations 1.4 and 1.5 respectively for every sample data provided to the Propagated Band Model (PBM) system.

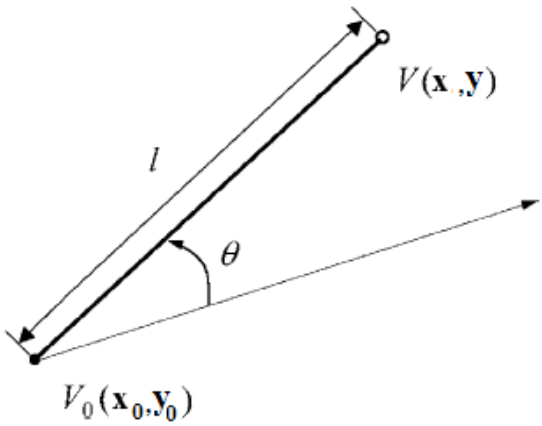


Figure 1.1: Line segment which constructs the network feature

$$x^{'} = x_0 + l^{'} \cos\theta^{'} \tag{1.4}$$

$$y^{'} = y_0 + l^{'} \cos\theta^{'}, \tag{1.5}$$

$l^{'}$ and $\theta^{'}$ are derived from the assumed true position of the line segment. Following this calculation, a vector of new end points are used during the derivation of error measures of the end point like σ_x^2, σ_y^2 and σ_{xy}. Band algorithm defined by Leung et al. (2004) is then applied to these constructed samples of new end point coordinates. For the following line segments, the same operation is applied but this time calculated end point coordinate is accepted as the start point coordinate of the

following line segment. This method considers the error which propagates from the start node of the previous line by applying this operation for each line of the network till the end node of the network. By the help of this approach, propagation of uncertainty is modeled for the whole network.

Line segment based uncertainty modeling algorithms like G-Band (Shi and Liu, 2000) and propagated error band (Leung et al., 2004) are applied to the network feature to compare the results of the proposed method.

G-band model is a probabilistic method which models the positional error of line segment for both independent and correlated cases of the two end points. G-Band model examines how the uncertainty in the location of a line's end nodes propagates along the line's interior. For this reason, generated band has a larger band width at the end points and it becomes smaller through the inner parts of the line segment. Distribution at the end points can be either equal or unequal to each other. This will affect the shape of the generated G-Band. G-Band views the error of each point along the line segment and models the quantity and direction of the error to indicate an error ellipse of the point. Finally composition of these error ellipses construct the G-Band.

Propagated error band model, which is proposed by Leung et al. (2004), is also a line segment based approach. It is based on basic measurement error model with the law of error propagation and models propagation of uncertainty from the start node of the line segment to the end node. The main idea is to define a functional dependency between the nodes of the line segment to model propagation of error in the start node. In this thesis, this functional dependency between the nodes of the line segment is constructed by using the distance and direction approach which is explained in Section 2.4.2.

Mentioned uncertainty modeling methods are then realized by the developed software application called Propagated Band Model (PBM) Drawer. Generated band geometries, derived statistics and comparison results are visualized through PBM Drawer by the help of GeoKIT Explorer which is a commercial GIS Java API.

Proposed method, which models the uncertainty propagation in a network of line segments, is important because modeling of uncertainty propagation in a network feature will form the basis of prediction of uncertainty in network based analysis techniques (i.e. route optimization, connectivity analysis, etc.) in GIS.

1.4 Thesis Organization

The thesis study is presented in logically distinct seven chapters, each of which contains some number of sections that are somehow related.

In this chapter, a brief introduction to the topic of error and uncertainty is provided and the aim of the thesis is described. Additionally, proposed method which is used to model uncertainty and its propagation through a network feature of line segments, is explained in detail.

The aim of the second chapter is to give the reader a general understanding of the uncertainty band models which are used in the literature and applied models which are extended within the thesis. Therefore, the second chapter starts with a detailed explanation of the line segment based uncertainty band models and continues with a general information about measurement error (ME) and covariance-based band generation process. At the end of the chapter, ME based band model which is the latest model for handling the uncertainty and its propagation in line segment based approach, is explained.

The third chapter attempts to explain the followed methodology within the thesis. Therefore, it is organized to give the reader a general understanding of the selected methods for data collection, statistical analysis and comparison of the results.

In Chapters 4 and 5, implementation of the thesis is given and developed software applications are explained in detail. In Chapter 4, previously examined band models for line segments are implemented as a plug-in application which is capable of generating an uncertainty region of given confidence values for a network feature of line segments. Other cooperative software applications are also explained within the content of this chapter. In Chapter 5, interactive application which is developed to generate a network feature of both linear and curvilinear geometries with simulated data provided by Monte Carlo Sampling, is explained.

The sixth chapter is organized to comment on the results of the applied band models. Analysis results are explained according to selected comparison parameters. Differences between the applied models are emphasized to clear up the concept. Finally, in the last chapter, analysis results are generalized and explained separately in the consideration of selected comparison parameter. The originality of the thesis is emphasized and possible future works and further improvements are discussed.

CHAPTER 2

ERROR ANALYSIS AND UNCERTAINTY PROPAGATION MODELS IN VECTOR-BASED GIS

As previously noted, spatial objects in vector-based GIS are comprised of points, lines, and polygons. Following sections aim to express the positional uncertainty models proposed for point, line and polygon features.

2.1 Uncertainty Models for Point Objects

Point errors are determined using standard deviations of x, y and z directional components of the point. Due to the two-dimensional nature of maps and imagery, error is quantified as a standard deviation in two directions (σ_x,σ_y) (King, 2000).

Uncertainty region of a point is visually represented by error ellipses and direction of the error ellipse is determined due to the dependency between coordinate pairs x and y. An error ellipse uses standard deviations in x and y directions to define semi-major and semi-minor axes and also the orientation of the ellipse (Wolf and Ghilani, 1997).

Thus semi-major, semi-minor and direction of the error ellipse can be derived as;

$$A^2 = \sqrt{\frac{1}{2}(\sigma_x^2 + \sigma_y^2 + w)} \tag{2.1}$$

$$B^2 = \sqrt{\frac{1}{2}(\sigma_X^2 + \sigma_Y^2 - w)} \tag{2.2}$$

and w is equal to;

$$w = \sqrt{(\sigma_X^2 - \sigma_Y^2)^2 + 4\sigma_{XY}^2} \tag{2.3}$$

and obliguity of the ellipse is derived as;

$$\tan 2\varphi = \frac{2\sigma_{XY}}{\sigma_X^2 - \sigma_Y^2} \tag{2.4}$$

Where A is semi-major axis of the ellipse, B is semi-minor axis and φ represents the obliguity of semi-major axis. σ_x^2 and σ_y^2 are variances of the x and y directional components of the coordinate which represents the point in interest. σ_{XY} represent the covariance of x and y. Components of the error ellipse is given in Figure 2.1.

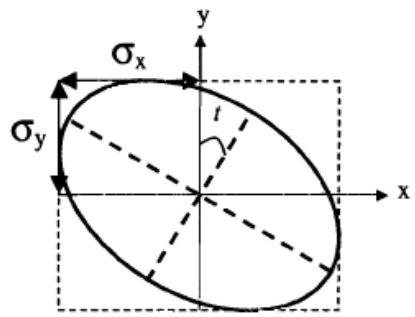


Figure 2.1: Error ellipse and its components (Adapted from King (2000))

In case of independent uncertainties which are equal in both directions ($\sigma_x = \sigma_y$), shape of ellipse is an error circle. Another measure for point uncertainty is Circular Error Probable (CEP) and is defined for probability of point's true position is lying inside a circle with radius CEP of % 50. Another measure for positional error is root-mean-square positional error (RMSP) which utilizes standard deviations (King, 2000). The RMSP for a point can be calculated by Caspary and Scheuring (1992);

$$RMSP = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{2.5}$$

2.2 Uncertainty Models for Line and Polygon Objects

Modeling the uncertainty of lines has an extensive history of epsilon bands (Caspary and Scheuring, 1992), concave curvilinear error bands (Dutton, 1992) and G-Bands (Shi and Liu, 2000). All these models focus on determination of uncertainty in the location of line's end nodes and propagates along the line's interior. Generated shapes

of these error bands are topologically identical and consider a region of uncertainty surrounding the entire line.

Chrisman and Perkal's epsilon band model is the earliest uncertainty model proposed by Perkal (1956, 1966) and Chrisman (1982). In this model, as lines are represented by a sequence of digitized points connected by straight segments, modeling error in the line is assumed to be derived from point's accuracy constructing the line. But this approach is inadequate for several reasons which are defined by (Goodchild et al., 1993);

- In digitizing process, digitizer operator tends to select points fairly carefully or select points those capture the form of line in a better way. It would therefore is not correct to regard the points are randomly sampled from line or regard the errors present in each point location is somehow typical of the errors that exists between true line and its digitized representation.

- The error between true and digitized lines are not independent and highly correlated.

Concept of error band which is proposed by Perkal (1956, 1966) is a method which is dealing with the specified problem. The model provides a boundary surrounding the line segment. It has been used both in deterministic and probabilistic forms. In the deterministic form, it is proposed that true line lies within the band with probability of 1.0 and never deviates outside. In the probabilistic form, the band is compared to a standard deviation or some average deviation value coming from the true line. It is assumed that any point selected over the line had a probability of 68% of lying within the band. Although Perkal's model is useful for describing errors for complex object, it has limitations for modeling probabilistic nature of the error.

Chrisman (1982) expands the work of Perkal (1966) and define a band, named *epsilon band*, which is based on a constant radius (epsilon) around the line's true or most likely position. ϵ calculated from the radius of the line's endpoint error circles, assuming a digitization process that yields random coordinate error in a circular normal distribution. The Figure 2.2 represents the shape of the band.

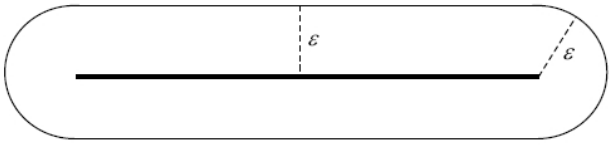


Figure 2.2: Chrisman's epsilon band (Adapted from Chrisman (1982))

The circular normal distribution is;

- two-dimensional,
- varies normally,
- have equal and uncorrelated errors in x and y directions.

The main drawback of the model is that it gives constant band width in the interior parts of the line. It gives no clear linkage with the positional errors of the end points constituting the line.

Dutton (1992) tested the Chrisman and Perkal's model with an assumption that digitizing points produces error that follows a circular normal distribution. Differing from others, Dutton's result describes concave curvilinear error band along the line. Figure 2.3 illustrates the experiment and results.

During the experiment a line is drawn and equal error circles of the endpoints are assigned for the true value. Endpoints are then generated from a circular normal distribution within each error circle and random endpoints are connected to form line segments. The distance from each segment to the median line is calculated at regular intervals along the line. The standard deviation for these distance residuals are then calculated at each interval along the median line. Consequently, Dutton finds that the displacement error from each segment to the median line is the greatest near the measured points and the least halfway between them. In conclusion, despite the precision of a segment's endpoint positions, its centerpoint is the most reliable location.

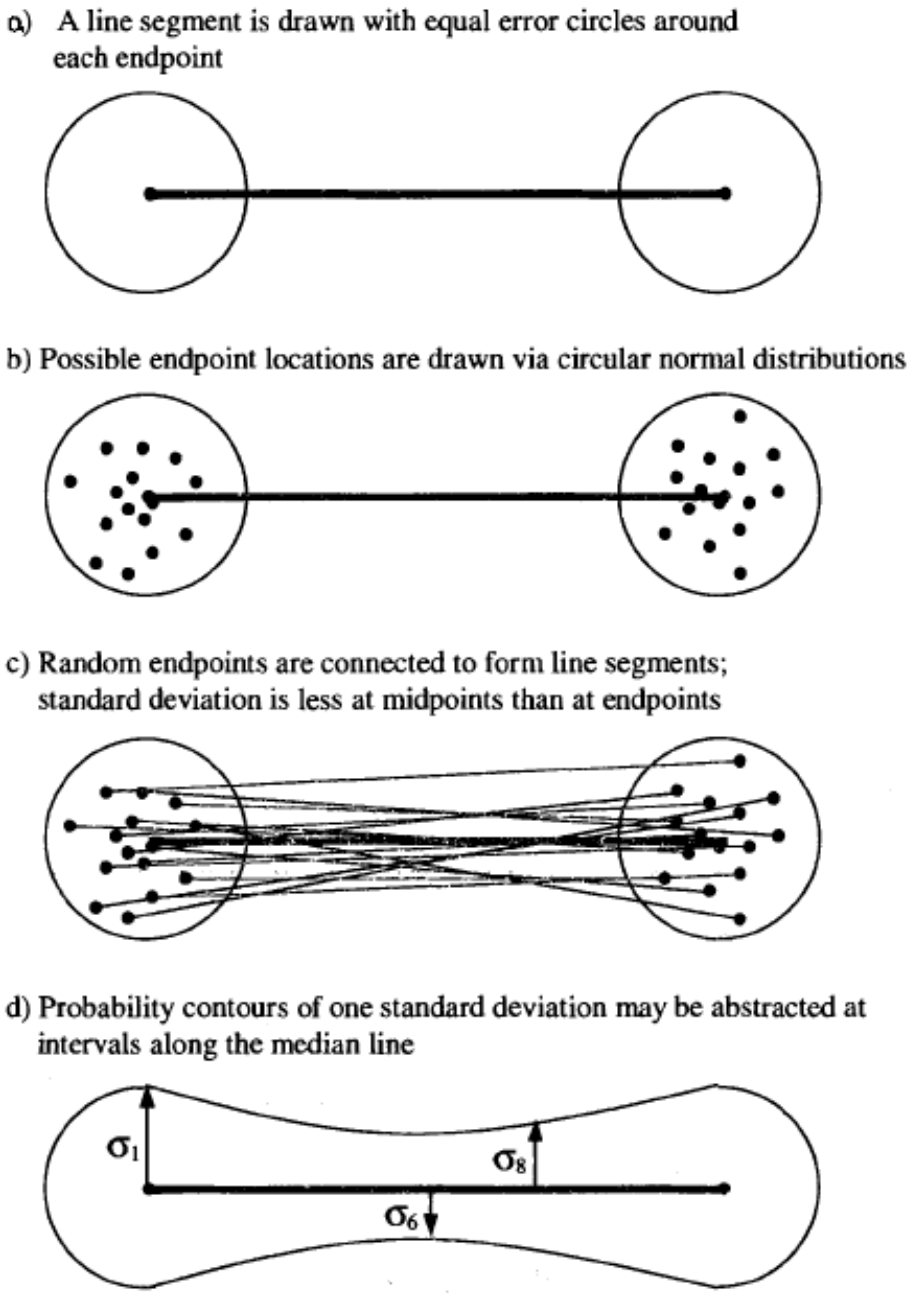


Figure 2.3: Dutton's Experiment simulating line segment (Adapted from Dutton (1992))

Caspary and Scheuring (1992) assumed that error at the end points are equal and follow a circular normal distribution. Like Dutton (1992) they use Monte Carlo simulation and error propagation to derive their error band which become narrow at the midpoint.

By applying error propagation law, and assuming that error at the end points of the line segment is independent and equal, they define the uncertainty variation along the line by equation 2.6:

$$\sigma_{x_t}^2 = \sigma_{y_t}^2 = \left(1 - \frac{2t}{L} + \frac{2t^2}{L^2}\right)\sigma^2 \tag{2.6}$$

where t is the distance between start point and the arbitrary point selected along the line, L is the total length of the line. Using equation 2.6 Caspary and Scheuring show that RMSP of the points in the middle of the line is less than the RMSP of end points and

$RMSP_t = \sqrt{2}\sigma_{x_t}$

As seen in Figure 2.4, mid point error is less than the end points with a factor of $\frac{\epsilon}{\sqrt{2}}$.

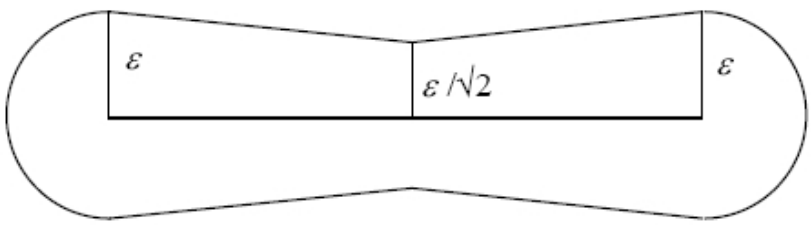


Figure 2.4: Caspary and Scheuring's Error Band (Adapted from Caspary and Scheuring (1992))

Although the inner error circles of the line is still independent from the end point error, this new approach is important for positional accuracy of objects. It is due to the fact that this approach uses error propagation law to derive the positional errors along the line and determines an error band by the error circle boundaries of all points along the line rather than error of the end points strictly (King, 2000).

In Shi and Tempfli's error band model, it is assumed that coordinate errors follow a normal distribution and are uncorreleated. Addition to previous models, error distribution of the end points effect the positional error and distribution of the selected point along the line, which means it is dependent on the errors of end points.

Shi (1994) define two endpoints as stochastic vectors following a normal distribution and define a four dimensional vector of equal variances and covariances which indicates the independence of end points. Shi (1994) derive the probability distribution of a point in a direction perpendicular to the line as seen in Figure 2.5 to investigate the marginal density of the line. Marginal density studies the point's distribution in a single direction within the two-dimensional distribution. This an indicator of realizing how the point's position can vary from its true value or mean position.

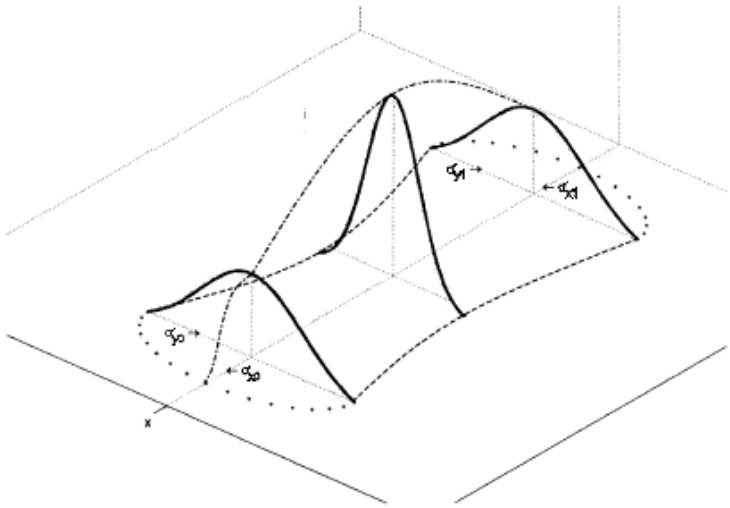


Figure 2.5: The probability density function of a line (Adapted from Shi (1994); Shi and Liu (2000))

Equations 2.7 and 2.8 give the standard deviations of end points along x axis which are equal like y axis such that $\sigma_{x_1} = \sigma_{x_2}$ and $\sigma_{y_1} = \sigma_{y_2}$. The error distribution of x and y axes of arbitrary point along the line segment are different from each other.

$$\sum_{p_1} = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1 y_1} \\ \sigma_{x_1 y_1} & \sigma_{y_1}^2 \end{bmatrix} \tag{2.7}$$

$$\sum_{p_2} = \begin{bmatrix} \sigma_{x_2}^2 & \sigma_{x_2 y_2} \\ \sigma_{x_2 y_2} & \sigma_{y_2}^2 \end{bmatrix} \tag{2.8}$$

Shi and Liu (2000)'s band model developes the idea stated by Shi (1994) and produce a general error band model, called *G-Band model*. Differing from the others, G-Band model considers the correlation between the end points, which is not handled in the previous band models. Also errors of the end points can be different from each other and follow two-dimensional normal distribution. They derive the errors of arbitrary points on the line to find out a distribution and density function for the whole line.

A line segment which is composed of two endpoints Z_0 and Z_1, as seen in Figure 2.6, can be represented as four-dimensional stochastic vector $Z_{01} = (X_0, Y_0, X_1, Y_1)$ which is a composite of two end points and follows a four dimensional normal distribution (Yu and Lu, 1983) with mean and variance-covariance matrix:

$\mu_{Z_{01}} = (mu_{X_0}, mu_{Y_0}, mu_{X_1}, mu_{Y_1})$

$$\Sigma_{Z_{01}Z_{01}} = \begin{bmatrix} \sigma_{X_0}^2 & \sigma_{X_0Y_0} & \sigma_{X_0X_1} & \sigma_{X_0Y_1} \\ \sigma_{Y_0X_0} & \sigma_{Y_0}^2 & \sigma_{Y_0X_1} & \sigma_{Y_0Y_1} \\ \sigma_{X_1X_0} & \sigma_{X_1Y_0}^2 & \sigma_{X_1}^2 & \sigma_{X_1Y_1} \\ \sigma_{Y_1X_0}^2 & \sigma_{Y_1Y_0}^2 & \sigma_{Y_1X_1} & \sigma_{Y_1}^2 \end{bmatrix} \tag{2.9}$$

The coordinates of an arbitrary point in the line segment can be represented by Equations 2.10 and 2.11 (Shi, 1994).

$$x(t) = (1-t)x_1 + tx_2 \tag{2.10}$$

$$y(t) = (1-t)y_1 + ty_2 \tag{2.11}$$

Where t is the point position fraction according to start point having a range of 0 and 1. To derive an uncertainty information matrix Σ_{ZZ} of a line as an extension of the covariance matrix of a point.

$$\Sigma_{ZZ}(t_1, t_2) = \begin{bmatrix} \sigma_X^2(t_1, t_2) & \sigma_{XY}(t_1, t_2) \\ \sigma_{YX}(t_1, t_2) & \sigma_Y^2(t_1, t_2) \end{bmatrix} \tag{2.12}$$

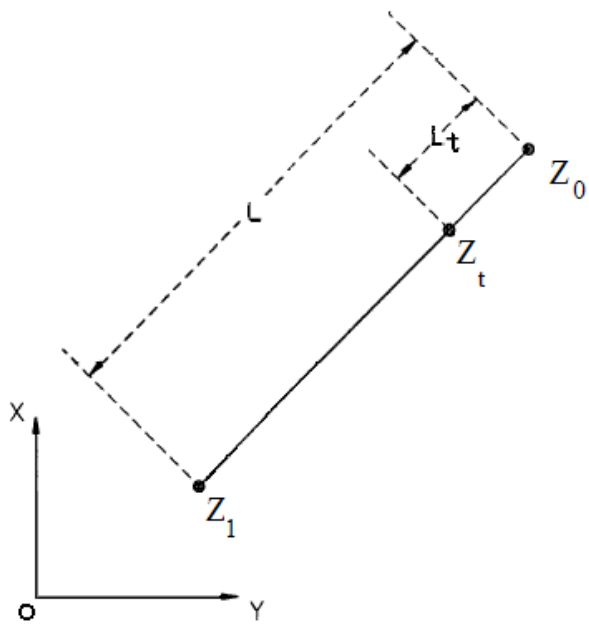


Figure 2.6: The line segment Z_0Z_1 is a composite of two end points $Z_0(X_0, Y_0)$ and $Z_1(X_1, Y_1)$ (Adapted from Shi and Liu (2000))

Equations 2.13 and 2.14 are variance functions of selected arbitrary point in x and y directions.

$$\sigma_x^2(t) = ((1-t)\sigma_{x_0} + t\sigma_{x_1})^2 = (1-t)^2\sigma_{X_0}^2 + 2t(1-t)\sigma_{X_0X_1} + t^2\sigma_{X_1}^2 \tag{2.13}$$

$$\sigma_y^2(t) = ((1-t)\sigma_{y_0} + t\sigma_{y_1})^2 = (1-t)^2\sigma_{Y_0}^2 + 2t(1-t)\sigma_{Y_0Y_1} + t^2\sigma_{Y_1}^2 \tag{2.14}$$

Covariance function is multiplication of functions of $\sigma_x(t) = (1-t)\sigma_{x_0} + t\sigma_{x_1}$ and $\sigma_y(t) = (1-t)\sigma_{y_0} + t\sigma_{y_1}$, respectively. The covariance equation is

$$\sigma_{xy}(t) = \sigma_{yx}(t) = (1-t)^2\sigma_{x_0y_0} + t(1-t)(\sigma_{x_1y_0} + \sigma_{x_0y_1}) + t^2\sigma_{x_1y_1}. \tag{2.15}$$

Similar to previous band models, the G-band's shape is described by the error ellipses of all points along the line segment. The error ellipses at the endpoints describe the ends of the the band, while the error ellipses of the inner points construct the boundary lines of the band (Figure 2.7). The error ellipses are described by their semi-major axis, semi-minor axis, and direction to the semi-major axis, defined in Section 2.1.

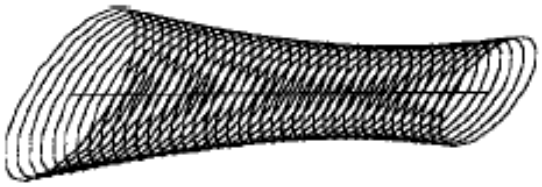
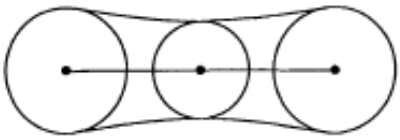
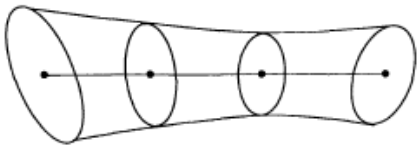


Figure 2.7: Visualization of G-Band from error ellipses (Adapted from Shi and Liu (2000))

Figure 2.8 illustrates the more general case of the G-band, which allows each endpoint to have varying errors in both dimensions. In this case, the endpoint errors in both directions are correlated and of varying magnitudes.



(a) G-Band for equal, uncorrelated errors



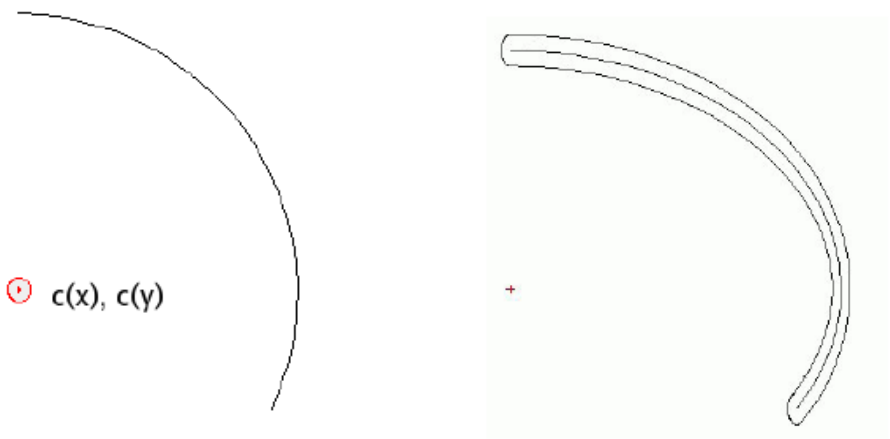
(b) G-Band for unequal, correlated errors

Figure 2.8: G-bands under varied conditions (Adapted from Shi and Liu (2000))

Kurtar (2006) extends G-Band model for functional curves which are generated by a non-linear function. In this study uncertainty models for arc, arc string, cubic spline and clothoid are proposed. Arc geometry is handled separately for arc by center point and arc by three coordinates. The first type is arcs by three coordinates which are defined by the set of coordinates of three points. The other type is arcs by center point which are defined by only one reference coordinate.

In Kurtar's study, the proposed uncertainty models for arc by three coordinates, arc string and cubic spline are based on G-Band model. The uncertainty models for arc by center point and clothoid are modeled by epsilon ellipse model. Kurtar (2006) proposes epsilon band model for arc by center point and clothoid because they are created with a single geographic coordinate together with other scalar variables. Hence, the band models of these functional curves use a unique epsilon ellipse for all of the arbitrary points over the curves. The other functional curves which are defined with more than one coordinate are based on G-Band model.

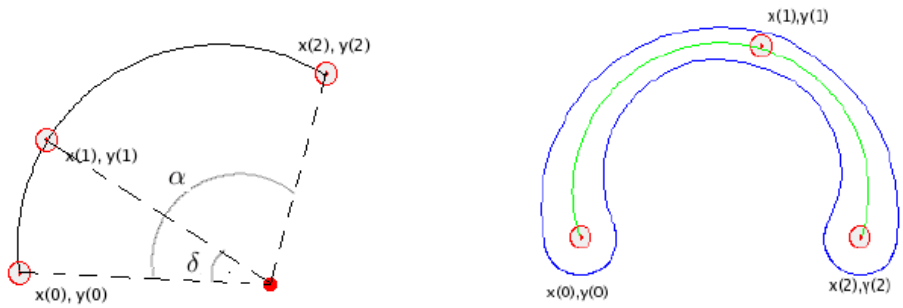
Functional geometries; arc by three coordinates, arc by center point and cubic spline in Figures 2.9(a), 2.10(a), 2.11(a) and proposed bands for these geometries can be seen in Figures 2.9(b), 2.10(b), 2.11(b), respectively.



(a) Arc By Center Point

(b) Proposed Band for Arc By Center Point

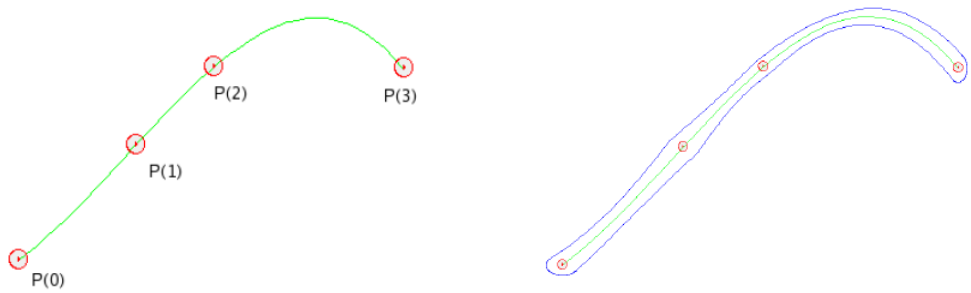
Figure 2.9: Arc By Center Point and Proposed Uncertainty Band (Adapted from Kurtar (2006))



(a) Arc By Three Coordinates

(b) Proposed Band for Arc By Three Coordinates

Figure 2.10: Arc By Three Coordinates and Proposed Uncertainty Band (Adapted from Kurtar (2006))



(a) Cubic Spline

(b) Proposed Band for Cubic Spline

Figure 2.11: Cubic Spline and Proposed Uncertainty Model (Adapted from Kurtar (2006))

In the literature, positional uncertainty in polygon object boundaries are modeled using previously defined line uncertainty models because line segment is the building block of line string which is used to express the boundary of polygon objects. As polygon boundary is a closed line string, line segment based uncertainty modeling can be applied on each line of the polygon.

For example, Mas (2006) applies epsilon band model to manage uncertainty in boundaries of sliver polygons which appear during the overlay operation of maps in GIS in order to enhance differences between the two dates. The generated band encloses a confidence region with a specific probability of including the true location of the boundary.

2.3 Basic Measurement Error (ME) Model

In this section, a statistical approach to measurement error analysis and error propagation is introduced by Leung et al. (2004), to realize MBGIS. In this approach, ME theory introduced by Neuilly and CETAMA (1999) and its analysis in GIS, (Heuvelink et al., 1989; Heuvelink, 1998), are extended to propose a rigorous model for estimation and propagation of locational errors. Proposed model is clearly explained in the following subsections including the basic measurement error model and error propagation.

As GIS is an abstraction of real world entities, GIS data is mainly provided by observations and measurements. In most of the cases, measurements are distorted by errors dependent on the selected method or device. Variations in GIS data due to the measurement error are introduced by faulty observation, biased observers, or by miscalibrated or inappropriate equipment. The reason is probably the natural variation in the data being collected. Additionally, processing error which is related to data manipulation such as digitizing and geocoding, overlay and boundary intersections, and errors from rasterizing a vector map, are some other sources of error which are stated by Burrough and McDonnel (1997). In Figure 2.12, operator error introduced during digitizing can be seen.

Measurement errors are generally introduced during data collection and are one of the main sources of uncertainty. Data collection can either be done through direct

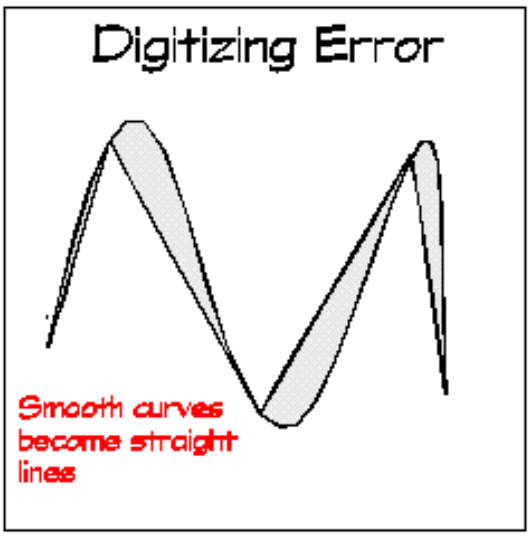


Figure 2.12: (Adapted from Montana_University (2007)) Operator Error During Digitizing Process

or indirect measurements.

The measurement error (ME) model for direct measurements are expressed by Leung et al. (2004) as;

$$x = \mu + \epsilon \tag{2.16}$$

where x is vector for the measurement value, μ is its true value vector and ϵ the vector of errors in measurements (ME vector). μ represents the true coordinate while x is the measured coordinates of a location.

As described in section 1.1, indirect measurements are found by using the mathematical dependence between measurand and its arguments, and the errors which are present in the original direct measurements are distributed during the computational process. By this way, errors existing in indirect measurements represents a functional view of the errors in original measurements. Therefore, distribution of errors from direct measurements via this functional dependency, which exists between direct and indirect measurements, are called *error propagation* (Wolf and Ghilani, 1997).

Mathematical representation of measurement error model for direct and indirect measurement conditions is given below (Leung et al., 2004): Leung et al. (2004) define $x \in R^p$ be the set of p measurements which are measured directly, $y \in R^q$ be q required quantities to be measured, and f be the function which connects the measurements

x to the quantities y as seen in Equation 2.17:

$$\mathbf{y} = f(\mathbf{x}) \tag{2.17}$$

If measured x and defining function f is known, y and the error distribution can be determined automatically. f^{-1} represents the inverse of f and it allows measurements x to be determined from the quantities y. Then basic ME model for indirect measurements are expressed by Leung et al. (2004) as,

$$(I) = \begin{cases} \mathbf{Y} = f(\mathbf{X}) \\ \mathbf{X} = \mu_x + \epsilon_x, \epsilon_x \sim (\mathbf{0}, \Sigma_x) \end{cases} \tag{2.18}$$

where **Y** the indirect measurement-value vector obtained by f from the random measurement-value vector, **X**. **X** is composed of the true-value vector μ_x, and the random ME vector ϵ_x with zero mean **0** and the variance-covariance matrix, Σ_x. According to Equation 2.18, variance-covariance matrix ϵ_y of **Y** is propagated from ϵ_x and also ϵ_y depends on the true-value vector μ_x. As a result, Leung et al. (2004) express the dependence model (I) as,

$$\mathbf{Y} = f(\mathbf{X}) = f\left(\mu_x + \epsilon_x\right) = E\left[f\left(\mu_x + \epsilon_x\right)\right] + \epsilon_y\left(\mu_x + \epsilon_x\right) = \mu_y + \epsilon_y \tag{2.19}$$

where $\mu_y \equiv E[f(\mu_x + \epsilon_x)]$ is dependent on ϵ_x and
$\epsilon_y \equiv \epsilon_y(\mu_x + \epsilon_x)$ is dependent on ϵ_x. Although both μ_y and ϵ_y is dependent on ϵ_x, the dependency of μ_y on ϵ_x is weaker than that of ϵ_y and should be omitted in general case because in most of the cases, this dependency does not occur. Therefore, $\Sigma_y \equiv cov(\epsilon_y) = cov[\epsilon_y(\mu_x + \epsilon_x)]$, and Σ_y can be defined as a function of Σ_x and μ_x, $\Sigma_y = F(\Sigma_x; \mu_x)$, which is dependent on μ_x. For example, if a function of $f(x)$ is defined on x as, $f(x) = a + Bx$, where a and B are a constant vector and matrix respectively, $\mu_y = a + B\mu_x$, $\epsilon_y = B\epsilon_x$, and $\Sigma_y = B\Sigma_x B^T$ can be expressed. This expression is a summary for error propagation based on functional dependence, and explained in the following subsection 2.3.1 in a detailed way.

Furthermore, as visualized in Figure 2.13, propagation of errors throughout any process can be obtained repeatedly by the help of the model (I), which is stated by Leung et al. (2004). For the given case, a vector $z \in R^r$ can be obtained by y and another known function $g(\cdot)$.

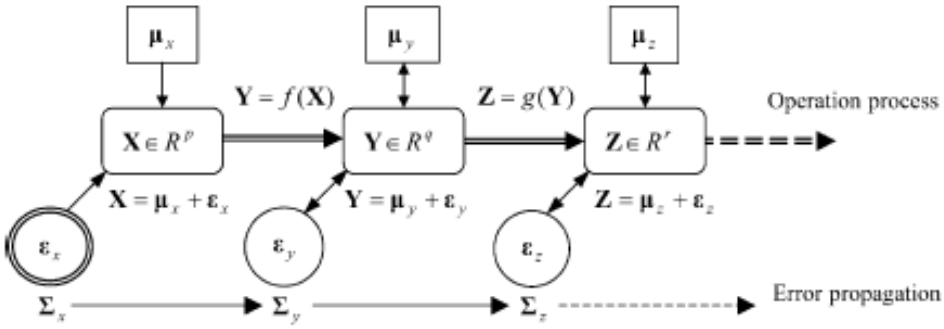


Figure 2.13: A flowchart for error propagation based on the bases ME Model (I) (Adapted from Leung et al. (2004))

2.3.1 Approximate Law of Error Propagation

As mentioned before, inaccuracy of indirect measurement result is expressed in the form of an uncertainty. Errors in measuring the position and attribute of a geographical entity or sampling GIS data generally trigger the occurence of uncertainty. Funtional dependency of indirect measurements to direct measurements force the input error present in direct measurements to propagate through expressed dependency to indirect measurement result.

According to Zhang and Goodchild (2002), uncertainty researches are concerned with identification, description and modeling of uncertainty and its propagation. Uncertainty propagation is predicting the effects on the analysis outputs using the given input uncertainty knowledge.

Mathematical representation of approximate error propagation law of Leung et al. (2004) determines error propagation theoretically in view of the transformation function $y = f(x)$.

In this case, two situations are considered by Leung et al. (2004):

1. The error distribution of Y which can be derived from the relationship $y = f(x)$.

2. The error distribution cannot be exactly obtained or is too complex to be derived because of nonlinearity of f.

In this subsection, derivation of an approximate law of error propagation for the second case, nonlinearity of function, can be seen.

For nonlinear case, the basic idea in the derivations of Leung et al. (2004) is to find out a linear (first-order) or second-order approximation to $y = f(x)$. Three different analytical approaches are offered to solve the GIS related problems and Heuvelink (1998) discusses on this topic. These approaches can be listed as;

- First and second order Taylor Series Approximation

- Rosenblueth's Point Estimate

- Monte Carlo Simulation

In the study of Leung et al. (2004), first-order Taylor Series Approximation is used to derive a general approximate equation for error propagation and transformation function of $y = f(x)$ is considerd in p variables $x = (x_1, \cdots, x_p)^T \in R^p$, where $y \in R$. This function is expanded into a Taylor series approximation about a point $a = (a_1, \cdots, a_p)^T \in R^p$ as given in Equation 2.20, in case of having derivatives up to $(n+1)$-order:

$$f(x) = f(x_1, \cdots, x_p) = \sum_{j=0}^{n} \frac{1}{j!} \left[\sum_{k=1}^{p} (x_k - a_k) \frac{\partial}{\partial x_k^{'}} \right]^j f(x_1^{'}, \cdots, x_p^{'}) \Big|_{x_1^{'}=a_1, \cdots, x_p^{'}=a_p} + R_n \tag{2.20}$$

where R_n is called the *remainder* after $n+1$ terms. So the first-order approximation of $f(x)$ is

$$\tilde{f}(x) = \tilde{f}(x_1, \cdots, x_p) = f(a_1, \cdots, a_p) + \sum_{k=1}^{p} (x_k - a_k) \frac{\partial f(x_1^{'}, \cdots, x_p^{'})}{\partial x_k^{'}} \Big|_{x_1^{'}=a_1, \cdots, x_p^{'}=a_p}, \tag{2.21}$$

or

$$\widetilde{f}(x) = c_a + b_a x, \tag{2.22}$$

where

$$c_a \equiv f(a_1, \cdots, a_p) - \sum_{k=1}^{p} a_k \frac{\partial f(x_1^{'}, \cdots, x_p^{'})}{\partial x_k^{'}} \mid_{x_1^{'}=a_1, \cdots, x_p^{'}=a_p}, \tag{2.23}$$

$$\underset{1\times p}{b_a} \equiv \left(\frac{\partial f(x_1^{'}, \cdots, x_p^{'})}{\partial x^{'} 1}, \cdots, \frac{\partial f(x_1^{'}, \cdots, x_p^{'})}{\partial x_p^{'}} \right)_{x_1^{'}=a_1, \cdots, x_p^{'}=a_p}, \tag{2.24}$$

c_a in Equation 2.23 and b_a in Equation 2.24 are independent of x and are respectively a constant and a constant matrix in these equations. Since 2.22 is an approximation to $y = f(x)$, it can also be represented as follows,

$$y \approx c_a + b_a x \tag{2.25}$$

Vector-matrix form representation of q functions $y_i = f_i(x)$ in $x = (i = 1, \cdots, q)$, which are approximated by the first-order Taylor expansion in Equation 2.22, is expressed by Leung et al. (2004) as:

$$y_i \approx c_{a,i} + b_{a,i} x, i = 1, \cdots, q, \text{i.e.,} \begin{pmatrix} y_1 \\ \vdots \\ y_q \end{pmatrix} \approx \begin{pmatrix} c_{a,1} \\ \vdots \\ c_{a,q} \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_{a,q} \end{pmatrix} x, \tag{2.26}$$

or

$$y \approx c_a + B_a x, \tag{2.27}$$

where $y = (y_1, \cdots, y_q)^T$, $c_a = (c_{a,1}, \cdots, c_{a,q})$, and the coefficient matrix

$$\underset{q\times p}{B_a} \equiv (b_{a,1}^t, \cdots, b_{a,q}^t)^T = \left(\frac{\partial f_i(x_1^{'}, \cdots, x_p^{'})}{\partial x_j^{'}} \right)_{x_1^{'}=a_1, \cdots, x_p^{'}=a_p}, \tag{2.28}$$

is called a *Jacobian matrix*. This is a matrix of partial derivatives with respect to each of the components.

Respect to these derivations, the approximate law of error propagation in the basic ME model (I) is given by Leung et al. (2004). They choose $a = \mu_x$ in Equation 2.27. Then for the random measurement vector, $Y \approx c_{\mu_x} + B_{\mu_x} X = (c_{\mu_x} + B_{\mu_x} \epsilon_x)$

Thus,

$$E(Y) \approx c_a + B_a \mu_x,$$

$$Cov(Y) = E(Y - EY)(Y - EY)^T \approx B_{\mu_x}[E(X - EX)(X - EX)^T]B_{\mu_x}^T = B_{\mu_x}\Sigma_x B_{\mu_x}^T$$

Therefore,

$$\boxed{\Sigma_y \approx \widetilde{\Sigma_y} \equiv B_{\mu_x}\Sigma_x B_{\mu_x}^T}. \tag{2.29}$$

Equation 2.29 represents the covariance matrix Σ_y, which can be approximately expressed by the covariance matrix Σ_x, and the Jacobian matrix B_{μ_x}. As visualized in Figure 2.13, the error in X is propagated to the error in Y, $\Sigma_x \rightarrow \Sigma_y$.

Leung et al. (2004) emphasize the important parts of the proposed algorithm among the others, for example component-wise derivation of Heuvelink (1998) and approximate notation given in the relation derived by Wolf and Ghilani (1997), as:

- The proposed model uses matrix representation which makes the derivation of approximate law of error propagation simpler.
- The model is an approximation for the nonlinear function f because Taylor Series first-order approximation is used in the derivation.
- The model is local because approximation is effective only in the neighbourhood of μ_x. This means that, if $f(x)$ is nonlinear on x, equation 2.22 or equation 2.29 has a good approximation to the left-hand side only when x is very close to μ_x.
- Differing from the other models, here the Jacobian matrix B_μ is appended with the subscript μ to reflect the dependence on the local point μ_x.
- Equation 2.29 is called the *approximate law of error propagation* for nonlinear f. And in general law of error propagation:

$$(II) : \sigma_y = \begin{cases} \approx \widetilde{\Sigma_y} \equiv B_\mu \Sigma_x B_\mu^T, \text{if } f(x) \text{ is nonlinear in } x, \\ = B\Sigma_x B^T, \text{if } f(x) \text{ is linear in } x, \text{i.e., } f(x) = a + Bx \end{cases} \tag{2.30}$$

- The second-order Taylor method in 2.20 can also be applied and will give more accurate results. But the complexity of the derivation increases according to the order of used Taylor method. For this reason first-order Taylor method is used during the derivation to reduce the complexity of the derivation.

2.3.2 Covariance Based Error Band

Over the last twenty years, numerious methods have been developed for modeling the positional uncertainty in spatial features. Some positional uncertainty models for points and line segments are detailly explained in the former sections of this chapter. Here is a simple and unified error band model which is strictly established from the concept of covariance by Leung et al. (2004). They establish an error band model from covariance concept and name it as *covariance-based error band*. In the following part of this subsection derivation of this error band model is explained in details.

Positional error of linear features can be calculated by either comparing matching points or matching lines of independent digital datasets because points are building blocks of lines (Niel and McVicar, 2002). Thus, "Uncertainty of a point can be derived from the covariance matrix associated with it and can be presented by an error ellipse (in 2-D) or an error ellipsoid. Since a line consists of points, uncertainty of any point on the line can naturally reflect uncertainty of the line" (Leung et al., 2004).

To satisfy the relation in Equation 2.18; Leung et al. (2004) define $X_i \equiv (X_{i1}, X_{i2})^T$ be the random vector, $\mu_i \equiv (\mu_{i1}, \mu_{i2})^T$ be the true vector, and finally $\epsilon_i \equiv (\epsilon_{i1}, \epsilon_{i2})^T$ be the ME vector of the endpoint coordinates of the line segment V_1V_2. Figure 2.12 represents the line segment V_1V_2 and any point $V^{'}$ which is selected on the line segment.

Leung et al. (2004) consider the 4×1 joint ME vector $\epsilon_{(2)} \equiv (\epsilon_1^T, \epsilon_2^T)^T$ and let its covariance matrix be $\Sigma_{(2)}$,

$$\Sigma_{(2)} \equiv cov(\epsilon_{(2)}) = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \Sigma_{ij} \equiv cov(\epsilon_i, \epsilon_j), i, j = 1, 2, \tag{2.31}$$

The $subscript_{(2)}$ in Equation 2.31 means there are two points which construct the line segment. Then 4×1 joint vectors: $X_{(2) \equiv (X_1^T, X_2^T)^T}$, $\mu_{(2)} \equiv (\mu_1^T, \mu_2^T)^T$ satisfy,

$$X_{(2)} = \mu_{(2)} + \epsilon_{(2)}, \epsilon_{(2)} \sim (0, \Sigma_{(2)}). \tag{2.32}$$

Hence, for any point $V^{'}$ selected on the line segment which is visualized in Figure 2.14, Leung et al. (2004) derived its coordinate vector $X^{'}$ and represented by the joint coordinate vector $X_{(2)}$ as:

$$X^{'} = tX_1 + (1-t)X_2 = D_t X_{(2)}, 0 \leq t \leq 1, \tag{2.33}$$

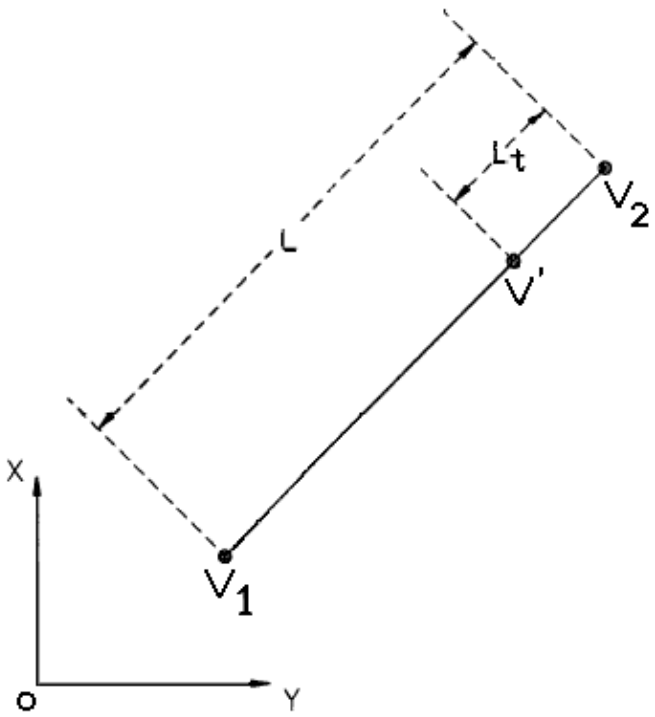


Figure 2.14: The line segment V_1V_2 is a composite of the two endpoints $V_1(x_1, y_1)$ and $V_2(x_2, y_2)$ (Adapted from Shi and Liu (2000))

where $D_t \equiv (tI_2(1-t)I_2)$ is a 2×4 constant matrix. Furthermore,

$$EX_{\prime} = tEX_1 + (1-t)EX_2 = t\mu_1 + (1-t)\mu_2 \equiv \mu_t, \tag{2.34}$$

$$\begin{aligned} cov(X^{'}) &= D_t cov(X_{(2)}) D_t^T = \begin{pmatrix} tI_2 & (1-t)I_2 \end{pmatrix} \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \begin{pmatrix} tI_2 \\ (1-t)I_2 \end{pmatrix} \\ &= t^2\Sigma_{11} + (1-t)^2\Sigma_{22} + t(1-t)(\Sigma_{12} + \Sigma_{21}) \equiv \Sigma_t \end{aligned} \tag{2.35}$$

For the case of independent end points, X_1 and X_2, the term $t(1-t)(\Sigma_{12} + \Sigma_{21})$ which indicates the covariation between the end point coordinates, will drop out automatically. And finally the Equation 2.34 can be reduced into:

$cov(X^{'}) = t^2\Sigma_{11} + (1-t)^2\Sigma_{22}$

2.4 Geodetic Model for Measurement Based GIS

Generally the GIS data are based on observations which are distorted by errors. Producer for these errors are usually the selected observation methods and these errors induce inconsistencies within the dataset. Hence, there exists a need for a general method which deals with these inconsistencies. According to the assumption that distribution of observation errors follows Gaussian distribution, these errors are modeled either by functional (deterministic) or statistical (stochastic) models (Navratil et al., 2004).

The idea under measurement-based GIS (MBGIS) is to store the original data rather than results of the statistical calculations. Therefore, no values are available for the errors of the measurements. Usually the results of the functional model are stored for the data set because the results of the statistical model involve data which needs large storage, i.e. variance-covariance matrix of the unknown parameters. For instance, MBGIS holds the coordinates and derives on the fly the error measures and statistical results. Thus if new measurements with higher quality are added to MBGIS, this approach improves the quality of the overall system (Navratil et al., 2004).

In the following sections a geodetic model, which is introduced by Leung et al. (2004) for measurement based GIS, is explained. Propagation of measurement error for line segments are also expressed in the following sections for various implementation types in geodesy.

2.4.1 Model Explanation

For the reason mentioned in the above section, MBGIS which is defined by Leung et al. (2004) provides access to the measurements (m), to the functions (f) and rules, rather than storing the results of the statistical calculations derived on these measurements. Then Leung et al. (2004) use these measurements to determine the object locations and use the functions and rules to determine the interpolated positions which are later used to derive the error propagation for the line segment. All these derivations are done on the fly and not stored.

After that Leung et al. (2004) constructed the MBGIS with reference to a geodetic model proposed by Shi et al. (1999). In this model, control points or monuments with

great accuracy are established and locations of the objects are indirectly measured from these monuments by an instrument. Figure 2.15 visualizes the hierarchy present within the model.

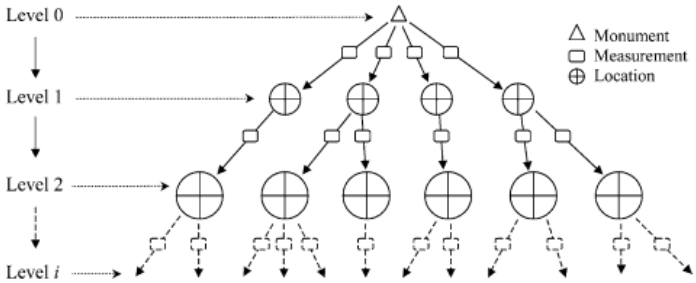


Figure 2.15: (Adapted from Shi et al. (1999)) Hierarchy tree of a simple geodetic model

At the top of the model control points are located. After that, location of the objects are established by measurements from these monuments. Since, there exists a strong correlation between the control points and the locations represented in the model, all points inherit the errors of the monuments. According to this hierarchy present between the monuments and the locations of the tree, Leung et al. (2004) derive the approximate law for error propagation at every location.

Based on the discussion above, Leung et al. (2004) structured a MBGIS as a hierarchy. According to the Figure 2.15, $x^{(i)}$ represents the location at level i. Then they derive the locations at level $i+1$ from locations at levels less than or equal to i through equation:

$$x^{(i+1)} = f_i(m^{(i)}, x^{(i)}). \tag{2.36}$$

Leung et al. (2004) use $x^{(0)}$ to represent the root of the tree which are monuments. At each level, the measurements $m^{(i)}$, the functions f_i are stored, and the locations $x^{(i)}$ are either stored or derived. After that by the help of relation between $x^{(i+1)}$ and $x^{(i)}$ depicted in 2.36, the propagation of the error present in the first level of the tree is modeled as follows:

- $M^{(i)}$; $\epsilon_m^{(i)}$: the random measurement vector
- $\epsilon_m^{(i)}$: the ME vector

- $\mu_m^{(i)}$: the true value vector of $\mu^{(i)}$
- $X^{(i)}$: the random measurement vector of $x^{(i)}$
- $\Sigma_m^{(i)} \equiv cov(\epsilon_m^{(i)})$

Then, Leung et al. (2004) express $M^{(i)}$ as,

$$M^{(i)} = \mu_m^{(i)} + \epsilon_m^{(i)}, \epsilon_m^{(i)} \sim (0, \Sigma_m^{(i)}). \tag{2.37}$$

$x^{(0)}$ is the monument, and viewed without ME. Unified notations are;

- $M_{(1)} \equiv M^{(0)}$
- $\mu_{(1)} \equiv \mu^{(0)}$
- $\epsilon_{(1)} \equiv \epsilon^{(0)}$
- $\Sigma_{(1)} \equiv cov(\epsilon_{(1)}) = \Sigma_m^{(0)}$

Thus the ME model for $x^{(1)}$ is summarized by Leung et al. (2004) as:

$$(M.1) : \begin{cases} X^{(1)} = g_1(M_{(1)}) \equiv f_0(M^{(0)}, x^{(0)}), \\ M_{(1)} = \mu_{(1)} + \epsilon_{(1)}, \epsilon_{(1)} \sim (0, \Sigma_{(1)}), \end{cases} \tag{2.38}$$

which is the basic ME model (I) given in equation 2.18. So, by similar procedure, the corresponding approximate law for error propagation is established by Leung et al. (2004) as:

$$\Sigma_x^{(1)} \equiv cov(X^{(1)}) \approx \tilde{\Sigma}_x^{(1)} \equiv B_{\mu_{(1)}} \Sigma_{(1)} B_{\mu_{(1)}}^T,$$

where $B_{\mu_{(1)}}$ is the Jacobian matrix of $g_1(M_{(1)})$ at $M_{(1)} = \mu_{(1)}$.

According to Equations 2.36 and 2.37, the composite function g_2 are obtained by Leung et al. (2004) as follows:

$$X^{(2)} = f_1(M^{(1)}, X^{(1)}) = f_1(M^{(1)}, f_0(M^{(0)}, x^{(0)})) = g_2(M_{(2)}),$$

where $g_2(M_{(2)}) \equiv f_1(M^{(1)}, f_0(M^{(0)}, x^{(0)}))$ defined by f_0 and f_1 is a function of the joint ME vector $M_{(2)} \equiv ({M^{(0)}}^T, {M^{(1)}}^T)^T$. Similarly, the symbols $\mu_{(2)}$ and $\epsilon_{(2)}$ are defined, and the ME model, like (I) in equation 2.18, for $x^{(2)}$ are obtained by Leung et al. (2004):

$$(M.2) : \begin{cases} X^{(2)} = g_2(M_{(2)}), \\ M_{(2)} = \mu_{(2)} + \epsilon_{(2)}, \epsilon_{(2)} \sim (0, \Sigma_{(2)}), \end{cases} \tag{2.39}$$

and the approximate law of error propagation is: $\Sigma_x^{(2)} \equiv cov(X^{(2)}) \approx \tilde{\Sigma}_x^{(2)} \equiv B_{\mu_{(2)}} \Sigma_{(2)} B_{\mu_{(2)}}^T$

where $B_{\mu_{(2)}}$ is the Jacobian matrix of $g_2(M_{(2)})$ at $M_{(2)} = \mu_{(2)}$, $\Sigma_{(2)} \equiv cov(\epsilon_{(2)})$, $\epsilon_{(2)} \equiv (\epsilon_m^{(0)^T}, \epsilon_m^{(1)^T})^T$, and $\epsilon_m^{(i)} (i = 0, 1)$ are defined by equation 2.37. The propagation relation between the error covariance matrices $\Sigma_{(1)} = \Sigma_m^{(0)}$ and $\Sigma_{(2)}$ is given by Leung et al. (2004) as follows:

$$\Sigma_{(2)} = \begin{pmatrix} \Sigma_{(1)} & \Sigma_{(1),m(1)} \\ \Sigma_{(1),m(1)}^T & \Sigma_m^{(1)} \end{pmatrix}$$

Recursively,

$$\begin{aligned} X^{(i)} &= f_{i-1}(M^{(i-1)}, X^{(i-1)}) = f_{i-1}(M^{(i-1)}, f_{i-2}(M^{(i-2)}, X^{(i-2)})) \\ &= \ldots = f_{i-1}(M^{(i-1)}, f_{i-2}(M^{(i-2)}, \ldots, f_0(M^{(0)}, x^{(0)}) \ldots)) = g_i(M_{(i)}), \end{aligned} \tag{2.40}$$

which results in the general ME model:

$$(M.i) : \begin{cases} X^{(i)} = g_i(M_{(i)}), \\ M_{(i)} = \mu_{(i)} + \epsilon_{(i)}, \epsilon_{(i)} \sim (0, \Sigma_{(i)}), \end{cases} \tag{2.41}$$

and

$$\Sigma_x^{(i)} \equiv cov(X^{(i)}) \approx \tilde{\Sigma}_x^{(i)} \equiv B_{\mu_{(i)}} \Sigma_{(i)} B_{\mu_{(i)}}^T, \tag{2.42}$$

where $B_{\mu_{(i)}}$ is the Jacobian matrix of $g_i(M_{(i)})$ at $M_{(i)} = \mu_{(i)}$, $\Sigma_{(i)} \equiv cov(\epsilon_{(i)})$, $\epsilon_{(i)} \equiv (\epsilon_m^{(i-1)^T}, \epsilon_m^{(i-1)^T})^T = (\epsilon_m^{(0)^T}, \ldots, \epsilon_m^{(i-2)^T}, \epsilon_m^{(i-1)^T})^T$, and $\epsilon_m^{(i)} (i = 0, \ldots, i)$ are defined by equation 2.37. The propagation relation between the error covariance matrices is given by Leung et al. (2004) as;

$$\Sigma_{(i)} = \begin{pmatrix} \Sigma_{(i-1)} & \Sigma_{(i-1),m(i-1)} \\ \Sigma_{(i-1),m(i-1)}^T & \Sigma_m^{(i-1)} \end{pmatrix}$$

Finally, the models $(M.i)$ and Equation 2.42 which are proposed by Leung et al. (2004) can be used to model the error propagation from the root $x^{(0)}$ to any location $x^{(i)}$ at level i.

Using the results above, Leung et al. (2004) investigate error propagation in three simple measurement problems in geodesy which are;

- Measurement with a distance and a direction

- Measurement with two distances
- Measurement with two angles

Within the subject of this thesis, measurement with a distance and a direction problem is applied to the line segments of the network to model line-based error propagation for every line. Former two problem approaches can be found in the study of Leung et al. (2004) with detailed explanation and examples.

2.4.2 Measurement with a distance and a direction

Leung et al. (2004) derive error propagation for a line segment which is measured with a distance and a direction or angle. In this section this derivation is expressed in a detailed way. Also this derivation is used, within the subject of this thesis, to implement a tool which is used to model the error propagation of a continuous line network defined in Section 3.1.1.

Reference line can be seen in Figure 2.16 with parameters defined by Leung et al. (2004) as follows:

- V_0 be the point with known coordinates vector x_0
- V the measured point with unknown coordinate vector x
- l the distance between V_0 and V
- θ the angle formed by the line segment V_0V

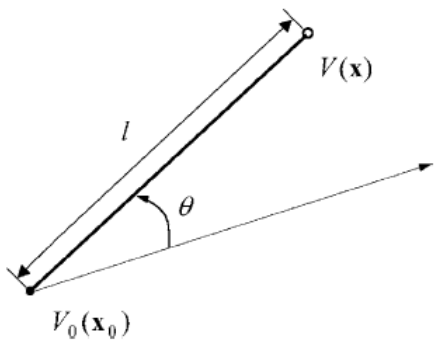


Figure 2.16: (Adapted from Leung et al. (2004)) Measurement with a distance and a direction

Main aim here is to see the propagation of measurement error when x is measured by the measurement $m \equiv (l, \theta)^T$ with the ME vector $\epsilon_m \equiv (\epsilon_l, \epsilon_\theta)^T$. At first transformation function $x = f(m)$ is defined to model the dependence between m and x. Expression is like:

$$\begin{cases} x_1 = x_{01} + l(\cos(\theta + \phi_0)), \\ x_2 = x_{02} + l(\sin(\theta + \phi_0)), \end{cases}$$

where ϕ_0 is a suitably selected constant such that $\theta + \phi_0$ becomes the angle with the x_1-coordinate axis. Thus by $x = (x_1, x_2)^T$,

$x = f(m) \equiv x_0 + l \cdot r(\theta), m \equiv (l, \theta)^T, r(\theta) = (\cos(\theta + \phi_0), \sin(\theta + \phi_0))^T$

Therefore, under the effect of ME ϵ_m, the observed coordinate vector X of the point V is represented by Leung et al. (2004) as a positional ME model like 2.18:

$$\begin{cases} X = f(M) \equiv x_0 + L \cdot r(\Theta), \\ M = \mu_m + \epsilon_m, \epsilon_m \sim (0, \Sigma_m), \end{cases} \tag{2.43}$$

where

- $\mu_m \equiv (\mu_l, \mu_\theta)^T$ are the true value vector of l
- θ, $M \equiv (L, \Theta)^T$ measured value vector in the presence of ME ϵ_m
- $\Sigma_m \equiv cov(\epsilon_m)$

Then Leung et al. (2004) define displacement on the line as follows:

$$\begin{aligned} dx = df(m) = (dl) \cdot r(\theta) + l \cdot dr(\theta) = r(\theta)(e_{2,1}^T dm) + l \cdot \\ (-\sin(\theta + \phi_0), \cos(\theta + \phi_0))^T d\theta \\ = [r(\theta)e_{2,1}^T + l \cdot (-\sin(\theta + \phi_0), \cos(\theta + \phi_0))^T e_{2,2}^T](dm), \end{aligned} \tag{2.44}$$

$$\begin{aligned} B_{\mu_m} &= [r(\theta)e_{2,1}^T + l \cdot (-\sin(\theta + \phi_0), \cos(\theta + \phi_0))^T e_{2,2}^T]_{m=\mu_m} \\ &= \begin{pmatrix} \cos(\mu_\theta + \phi_0) & -\mu_l \sin(\mu_\theta + \phi_0) \\ \sin(\mu_\theta + \phi_0) & \mu_l \cos(\mu_\theta + \phi_0) \end{pmatrix} \end{aligned} \tag{2.45}$$

$X = f(\mathbf{M})$ is nonlinear in $\mathbf{M}$. Therefore, Leung et al. (2004) obtain the approximate law of error propagation for the point V as:

$$\boxed{\widetilde{\Sigma}_x = B_{\mu_m} \Sigma_m B_{\mu_m}^T}. \tag{2.46}$$

Σ_x is the the covariance matrix which gives the ME of coordinates of the position V.

The Equation 2.46 enlarges with the addition of Σ_0, which represents the random ME that exists at the known point V_0. This error is independent of the measurements of l and θ. Then, from Equation 2.43, Equation 2.46 becomes (Leung et al., 2004);

$$\boxed{\widetilde{\Sigma}_x = \Sigma_0 + B_{\mu_m} \Sigma_m B_{\mu_m}^T}. \tag{2.47}$$

CHAPTER 3

METHODOLOGY

To explain the subject studied throughout this thesis, methodology section is divided into three parts. The first part, explains the data collection methods:

- Network Digitized by User (NDU), and
- Network Created by User (NCU)

The second part covers data analysis and defines the derived statistics and used band models to vizualize the uncertainty which propagates through the network feature. Uncertainty band models used during data analysis part are;

- G-Band with line segments
- Propagated Error Band
  - Line Segment-Based implementation
  - Network-Based implementation
- G-Band with Functional Geometries

Last part of the methodology compares the results of applied models according to:

- Band areas
- Intersection of band geometries
- Sample geometry coverage

The overall methodology followed in this thesis is given in Figure 3.1.

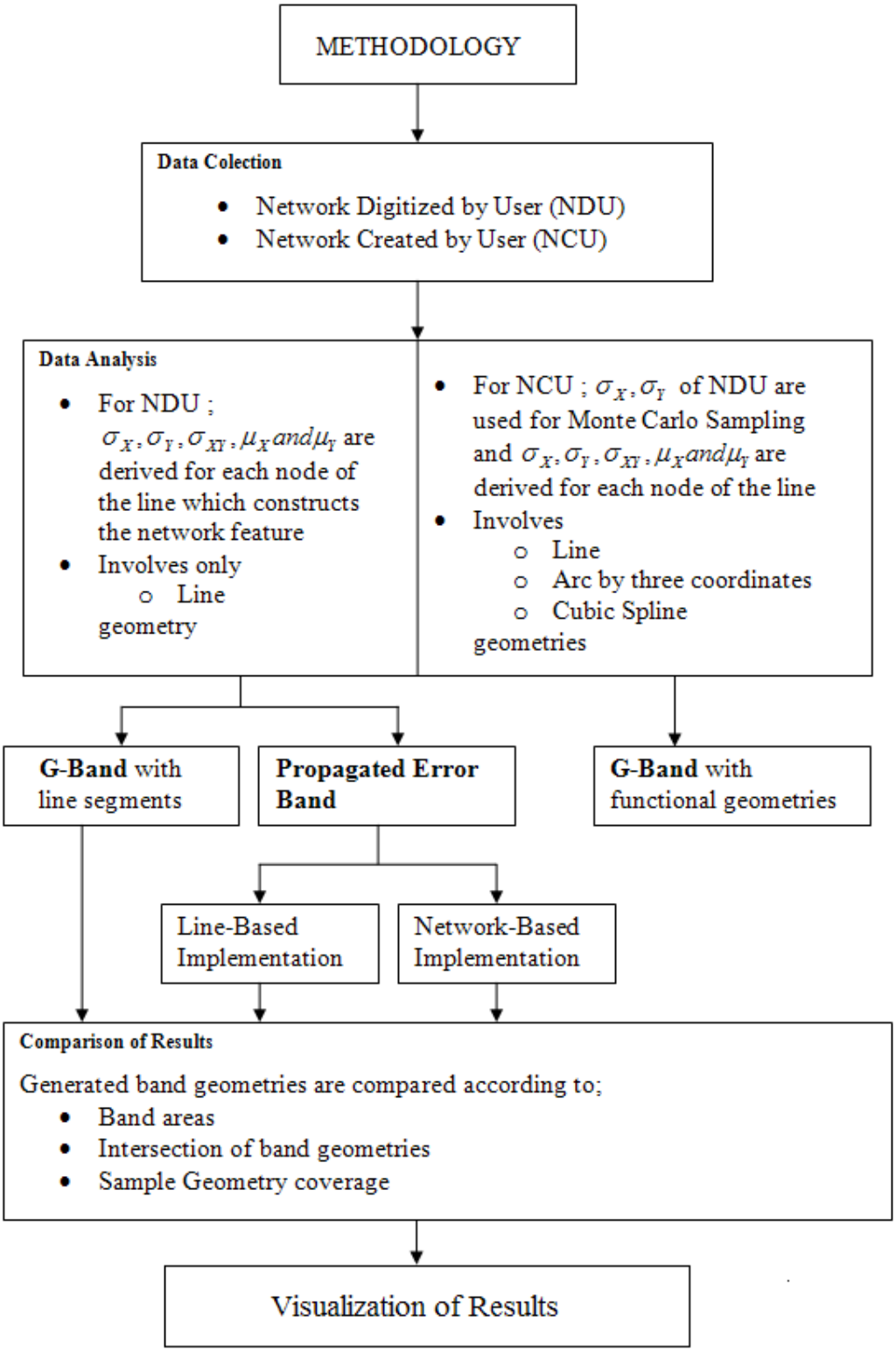


Figure 3.1: Methodology of the thesis

3.1 Data Collection Methods

The main reason for using different data collection methods is to reveal the difference between generated bands in cases of human digitized lines and simulated lines of network.

3.1.1 Explanation of the Selected Line Network

Selected linear network is a part of the road network in Middle East Technical University (METU). Figure 3.2 illustrates the selected linear network and its location within the METU campus.

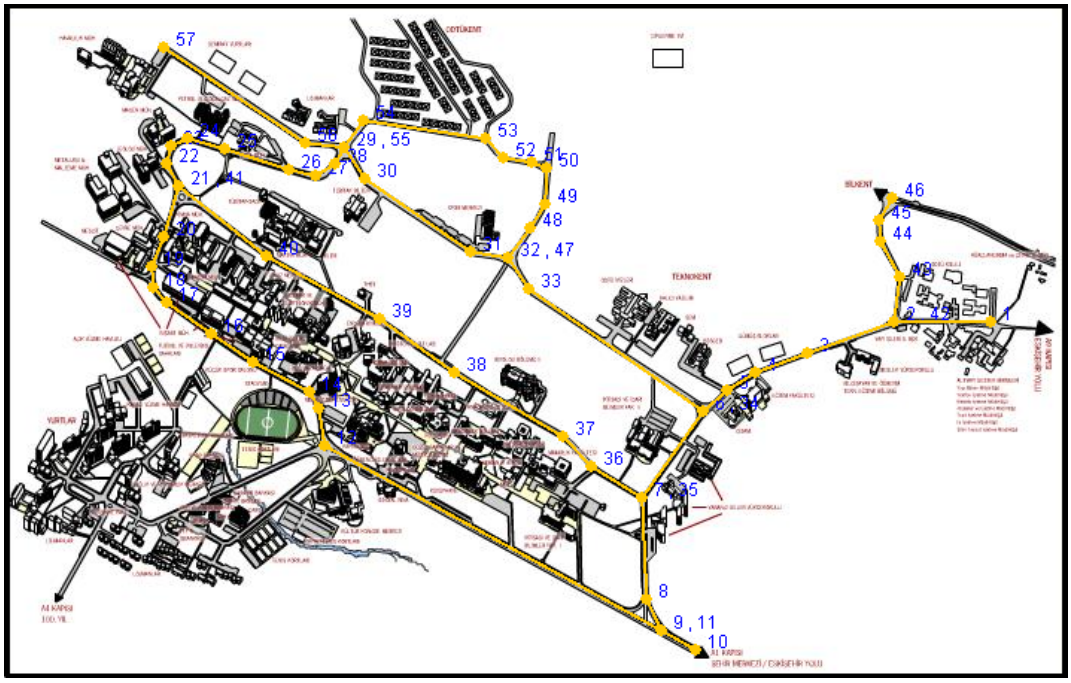


Figure 3.2: Selected Linear Network

As seen in Figure 3.3, specified line network has a start and an end node. Each node is defined by numbers which clarify the direction of the network. Start node is represented by number 1 and end node is represented by number 57. Some sharing nodes are represented by more than one numbers, because these nodes are used by two or more lines. To clarify the direction of the network within the data, each node is represented by different node number. Nodes which are shared by other line segments save the node id of these segments as reference node. To ensure the conti-

nuity of the network, sharing nodes are matched by the digitizing software which is explained in Section 4.2. Developed software compares the new coming coordinate with the previously entered one and if there exists a match within the data set then new node is replaced with the matching one.

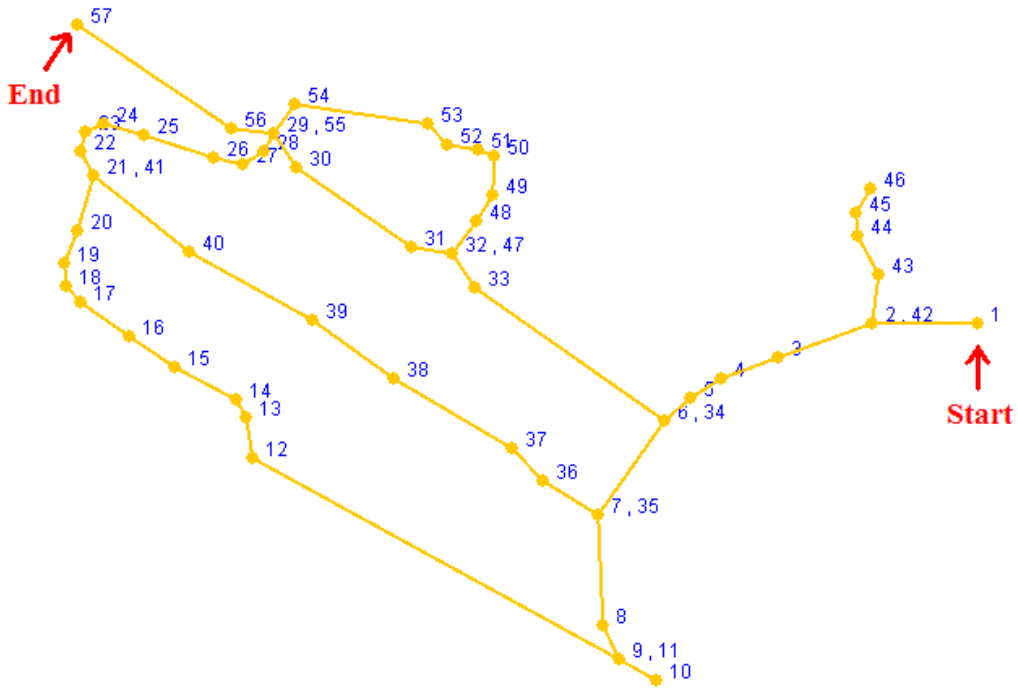


Figure 3.3: Selected Linear Network with Node Ids

As defined by Rodrigue (2007), network topology is the study of the arrangement or mapping of nodes and links in a network. They are the representations of location, direction and connectivity. If there is a break in the topology, entire network shuts down. In summary, selected network must have a *starting point* and an *ending point*. Selected line pieces which construct the network must be *solidly connected* to each other to secure the continuity.

3.1.2 Network Digitized By User (NDU)

Selected network that is described in Section 3.1.1 is asked to be digitized by the staff of both Geodetic and Geographic Information Technologies Department of Middle East Technical University and the firm Bilgi Geographic Information Conversion and Management Systems. Samples from 15 people are used during the analysis. Devel-

oped software is designed to generate results independent of the initial sample size; so if more samples are provided than results can be produced for the given set.

NDU data is collected from people during the digitization process which is simulated by application seen in Section 4.2. Sample Collection Applet is developed to serve an environment for digitizing process. Collected samples are then exported as text formatted file which are used by PBM Drawer Application

NDU aims to realize the real world digitization and try to find out which uncertainty model handles the personal error condition best. The second aim is to mark the contribution of errors related to the digitized line length and curvature.

3.1.3 Network Created By User (NCU)

In NCU case, user can define a network which can both be composed of linear and curvilinear geometries like; line, arc by three coordinate and cubic spline. Monte Carlo sampling is then used to generate a sample data for defined geometries within the network feature. Calculated variances on x and y directional components for NDU case is used during Monte Carlo sampling process and generated sample set is an input to PBM Drawer Application.

The coordinates which construct the supported geometry types, are sampled by method public **double** *nextGaussian()*

which generates a random number having mean of 0.0 and standard deviation of 1.0. It is the method of class java.util.Random which is in Java SDK(Standard Development Kit). Seed which is used during random number generation is produced by class java.security.SecureRandom (see Appendix C.4).The sampled lines are based on normal distribution.

ID application is developed to serve an environment for user to define a network interactively. It also visualizes the sampled lines or curves to inform user about the process. Detailed explanation of the application can be found in Chapter 5. Finally, generated sample set and calculated statistics are written to a file named *statistics file*. Types and organization of the statistics files are explained in Section 4.4.1.2 in detail.

3.2 Data Analysis Methods

Three different models are actively being used during the study held for the thesis and therefore in the developed applications. First one is G-Band model which is proposed by Shi and Liu (2000), and is a generic model which describes the positional error of line segments with an assumption that end point errors of the line segment follow two-dimensional error distribution.

The second model is also a covariance-based error band model which is proposed by Leung et al. (2004). Main difference between this error band model and the G-band model appears at the point of error propagation modeling. The model of, Leung et al. (2004), models the propagation of error at the start node of the line segment by defining a function based approach. The idea is defining a function for end point of the line segment by using start point related attributes. This functional representation is based on approximate law of error propagation and functional dependency can be presented by using distance and direction approach which is explained in Section 2.4.2. In this thesis, functional dependency for line segments are created using distance and a direction approach. The implementation of the algorithm is divided into two sub-parts. In the first part, algorithm is applied to each line segment constructing the network, separately to induce line based behaviour of the model. Another aim is to compare the results with G-band approach, which is also a line segment based model implementation. In the second part, a new method is defined to implement the uncertainty propagation model of Leung et al. (2004) to whole network. This method derive the end point of the line again from the provided statistics of start point of the line segment. This new end coordinate is used as the start node of the following line and the same process is repeated for the remaining lines of the network.

The third model is proposed by Kurtar (2006) to model uncertainty of functional curves like; arc by three coordinates, arc string and cubic spline. This model is an extension of G-Band model for functional curves which are defined with more than one coordinates. Although results of this model is not compared with previously mentioned ones, functionality of generating an uncertainty band for curved geometries is also implemented.

3.3 Comparison of Results

This is the last part of the methodology, which comments on the outcomes of the explained methodology. Comparison is applied between;

- G-Band with line segment and Line Segment-Based approach of Propagated Error Band,
- G-Band with line segment and Network-Based approach of the Propagated Error Band.

This comparison is made by using band area, band geometry intersections and sample geometry coverage. Chapter 6 includes the outcomes and stated conclusion for the comparison methods. The basic aim of these approaches and comparison methodology is to visualize the difference between error propagation for line segment based implementation and for the whole network feature.

3.4 Visualization of Results

The methodology explained is implemented in the base application called PBM Drawer System. It is a wizard implementation which can be used to visualize uncertainty and propagation of uncertainty in a line network. It works on a file based structure and can import files of specified type and organization in Section 4.4.1. Detailed information about system organization and interaction with other developed software applications, coding specifications and user manuals are given in Chapter 4.

ID application is developed to implement an environment to produce randomly created data for PBM system. It is a simulation for digitizing process and is explained in Chapter 5.

During the implementation of these applications Java is used. Also a commercial GIS toolkit named GeoKIT is used to ensure GIS capability. Finally, U-Bag API which was written by Kurtar (2006), is used during band generation process of functional curves.

CHAPTER 4

UNCERTAINTY BAND DRAWER SOFTWARE

In this chapter, details about the developed **P**ropagated Uncertainty **B**and **M**odel Drawer Software (PBM), which is written as a plug-in to a commercial GIS toolkit GeoKIT Explorer, is given in detail. Also some other cooperative software implementations are explained, like 'Data Collection Applet' and 'Measured and True Value Comparator Plug-in'. General requirements, inputs and outputs, and structure of the system are presented in details included with the code samples.

4.1 Implementation Overview

GeoKIT Explorer is a commercial software developed by Bilgi Geographic Information Conversion and Management Systems. It is a Java API (Application Programming Interface) for GIS and can be used for manipulating, visualizing, and analyzing 2D/3D raster and vector spatial data. It is written completely in Java programming language and provides a comprehensive set of components to embed GIS functionality into applications. GeoKIT is a platform independent product and implements the necessary functionality via separate modules. Each developed module serves different functionality to the overall system. Data access mechanism is served by Accessor architecture. OGC compliant web services, WFS(Web Feature Service) and WMS(Web Map Service), implement client and server side functionality. GeoKIT MTG Overlay Editor is an overlay and symbology editor application which can be used to manipulate C2 Tactical Graphics objects as specified by MIL-STD-2525B (Bilgi_GeoKIT, 2007).

GeoKIT Core and GeoKIT Explorer modules are used in the implementation of the thesis and supply necessary GIS components and GIS functionality. GeoKIT Explorer's Plugable API architecture which serves a high level plug-in API, facilitates

the integration of the developed application piece to GeoKIT. Additionally, GeoKIT Explorer Desktop Application's easy to use and friendly user interface smartens up the representation of the thesis results.

U-Bag is another API which is used during implementation of the PBM application. This API serves uncertainty band generation functionality for line, arc by three coordinates, arc string and cubic spline using G-Band model described in Chapter 2. And for arc by center point and clothoid, it uses epsilon band model. This API is either developed by using Java and GeoKIT Explorer.

Architectural relation between used and implemented modules can be seen in Figure 4.1. As seen, JVM (Java Virtual Machine) is the base and GeoKIT API, U-Bag API, PBM API and the PBM application runs over this substructure.

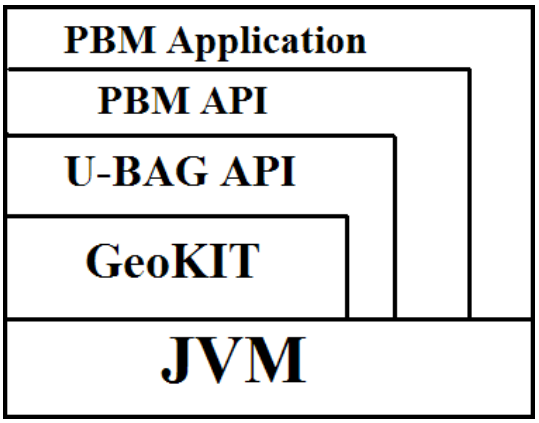


Figure 4.1: System Architecture

Implementation of the PBM Uncertainty Band Drawer Software can be divided as:

- PBM Propagated Uncertainty Band Generator API,
- PBM Propagated Uncertainty Band Generator Plug-in Application.

The sample data collection applet, Interactive Drawer(ID) Plug-in and sample (measured) and true value comparator plug-ins are explained in the following Section 4.2, Chapter 5 and Section 4.3, respectively.

Interaction between these plug-ins and the end user can be seen in Figure 4.2.

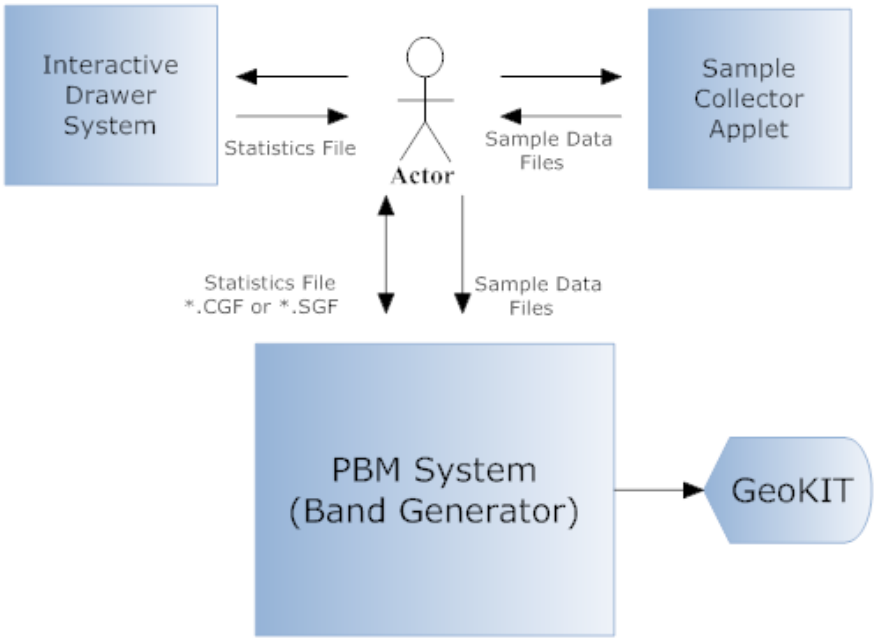


Figure 4.2: Interaction of user with implemented systems

4.2 Sample File Construction Software

Input data introduced to the system is provided from people who are asked to digitize the selected road network explained in Chapter 3. Data Collector is a java applet which is written to create sample files by using the digitizing data input. Data Collector applet can be seen in Figure 4.3. Data Collector applet has three separate parts;

1. Drawing Area,
2. Control Panel, and
3. Output Area.

Drawing area displays the vector data (lines and vertices) which form the selected network, standing over the image representation of the interest area. User can digitize over the presented vector data by using mouse events. During digitization process a guide line, as shown in Figure 4.3 with red color, orients the user. Each pointed coordinate is displayed on the 'Coordinate (x,y)' label of the control panel and defined line is also listed in the output area of the applet. During the drawing process Java Graphics2D class and its functionality is used.

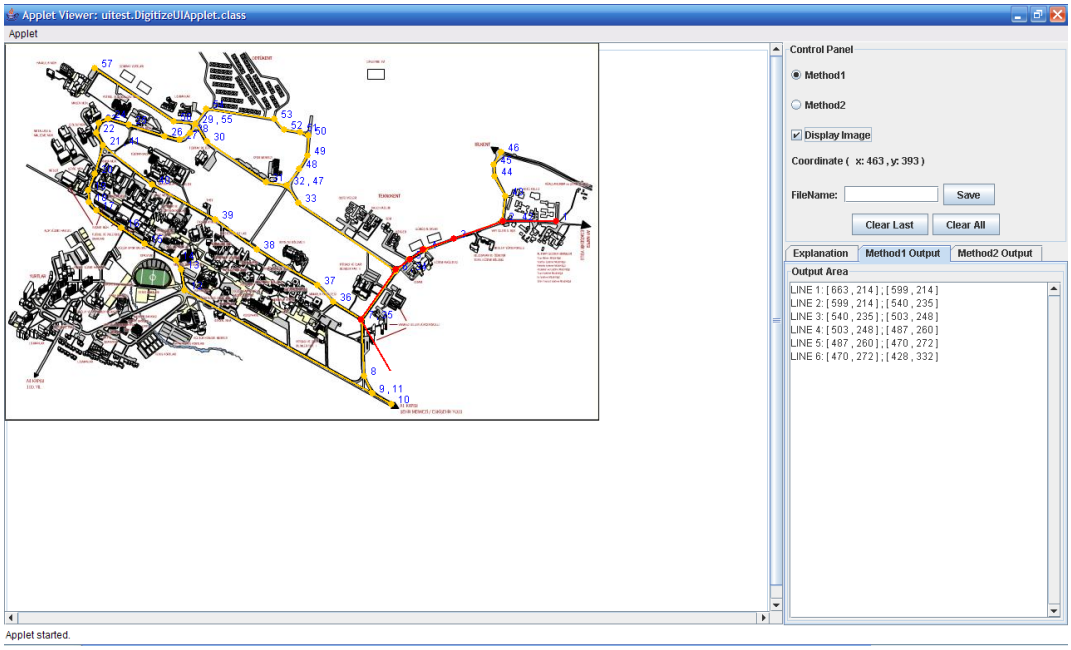


Figure 4.3: Sample Data Collection Applet

Control panel serves user some control parameters like, image on/off choice, pointed coordinate label and finally functionality served to save data into file. In output area, start and end coordinates of digitized line segment are listed. Finally the generated digitizing samples are saved with specified file name to a predefined location. Internal structure of the sample files are explained in subsection 4.4.1.1.

4.3 Sample Comparator Plugin

Main aim of this plug-in is to visualize the content of the sample files separately for being aware of the differences between provided assumed true value or mean of the samples and digitized sample lines of the network. This tool is also used during the comparison of the results generated by PMB Plugin Application. Graphical representation of plug-in can be seen in Figures 4.4 and 4.5.

Comparator plug-in's data selection dialog is used to introduce the directory of digitized samples and file is assumed to contain true values for the selected line network, if exists. Name of the file for provided assumed true values is, **base.txt** and data inside is organized with the specified file format given in section 4.4.1.1.

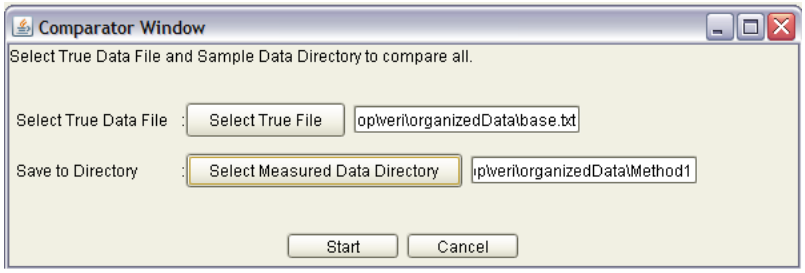


Figure 4.4: Dialog to introduce measured and true network file

After pressing the *Start* button, all files are visualized on the mapview of GeoKIT Explorer and user can select and compare each sample file with true geometries diplayed in blue color. See Figure 6.3.

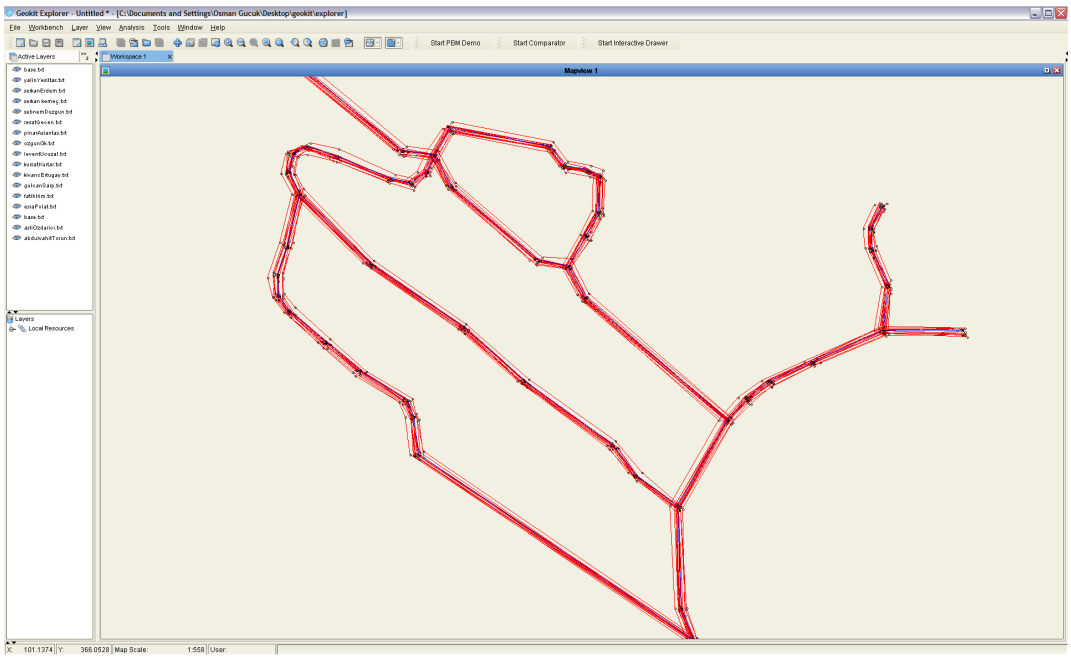


Figure 4.5: Measured and true data comparison plug-in

4.4 Uncertainty Band Generator Wizard

Nearly for 20 years researchers and acedemicians have been working for modelling uncertainty in GIS features and operations. Depending on these researches, several different algorithms are developed for point, line and polygon objects which are used to simulate real world entities in GIS. The algorithms which are frequently used in

the determination, derivation and visualization of uncertainty, are broadly explained in Chapter 1. Although, effective uncertainty modelling algorithms have been running in the literature, most of the GIS software developers do not give necessary importance on error and uncertainty modelling.

In this part of the thesis, algorithm defined by Leung et al. (2004) is implemented to model uncertainty and its propagation through a line network. Formulation and specifications are addressed in Chapter 2. This chapter focuses on the implementation of this algorithm considering two main selective cases. In the first selection, both covariance-based error band and G-band model algorithms are applied on sample data input. In the second selection, band generation process is handled from a file, called *statistics file* for its statistical data content. This file is generated by either Propagated Band Generation (PBM) Software by applying computations over sample data, or by Interactive Drawer (ID) Software which is explained in Chapter 5.

In the following sections, structure of the files which are used within the system, graphical user interfaces and workflow of the system are elucidated.

4.4.1 File Structure

PBM Drawer and other co-operating plug-ins are based on a file dependent I/O (input and output). PBM System accepts punctuation separated files as an input and also generate output files in the same format. Files hold the line network geometry data without using topological relationships. Each line of the file represents only one geometry. The internal structure and stored attributes vary according to the type of the file. Organization of sample and statistics files are clarified in the following sub sections.

4.4.1.1 Sample File Structure

Sample files are generated by Sample Collector Applet, explained in Section 4.2. Generated sample files include fields;

- Group ID: If the network is divided into subgroups which are representing a common property, then group id is assigned an integer value. For example, roads of the same district may have the same group id.

- Feature ID: Feature id is used to represent the feature uniquely within the whole network.

- Feature Type: Represents the type of the feature. Three different geometries are supported by PBM System; line, arc by three coordinates and cubic spline, but as the sample collector applet is designed to handle only line geometry, feature type of files generated from sample collector applet is equal to 'LINE'.

- Start Node ID: Line is defined by using two coordinates, first one represents the line's start and other represents the end. In this field, id of the start node is given.

- Start Node X: X directional component of the coordinate representing the line start.

- Start Node Y: Y directional component of the coordinate representing the line start.

- Reference Node IDs for Start Node: If this coordinate is shared by another line(s) then id representing the start or end node of the shared line is stored as referenced node id within the content of this field.

- End Node ID: Id of the end node which finalizes the line segment.

- End Node X: X directional component of the coordinate representing the end of line.

- End Node Y: Y directional component of the coordinate representing the end of line.

- Reference Node IDs for End Node: Referenced node id(s) which is/are sharing this end node.

All these fields are written to a file in a punctuation separated format by handling each line separately . Differing from the others, reference node ids are represented with enclosed paranthesis ' (' and ') ' and separated by semi colon ' ; '. An example for sample data file format can be seen in Appendix A.1. Sample files are written into a text document which has an extension of '*.txt'.

Sample file reader handles reading operation and represents the file content as a collection of Java objects which implement the interface *IPBMNetworkFeature*. This interface is implemented by all supported geometries like; line, arc by three coordinates and cubic spline. Hence each member of this interface presents three methods given in Appendix B.1:

int getGroupID(): returns the group id of the geometry, if exists.

int getFeatureID(): returns the id of the geometry.

String getFeatureTypeStr(): returns the string representation of geometry type. Possible selections are defined as static final attributes within the *IPBMNetworkFeature* class by considering the supported geometries. For line geometry it is 'LINE'; for arc by three coordinates, it is 'ARC' and finally for cubic spline it is defined as 'SPLINE'.

PBMSampleFileReader class is constructed by a directory information and collection of selected sample file names. Then *PBMSampleFileReader*'s *readSampleFiles()* method connect to given directory path, select and read the given sample file. Finally, *PBMSampleFileReader* class contains a vector of imported sample file's content vector called *sampleFilesVector*, in which each line is represented by a IPBMNetworkFeature class implement, (i.e. *PBMSampleLine*, *PBMSampleArc* and *PBMSampleSpline*). Also according to the incoming sample file content, file is categorized as simple or complex. Simple geometry files has only line geometry inside but complex geometry files involves other supported geometries. A boolean variable *simpleFileGeometry* of *SampleFileReader* class represent the content type of the sample files. If it is TRUE then it indicates that the content of the sample file is simple, otherwise it is complex.

4.4.1.2 Statistics File Structure

Another file format, accepted by PBM Drawer and generated by its co-operating plug-ins, is statistics file. This file is also represented in a punctuation separated way but it is categorized and named as;

- *Simple Geometry File, SGF* or
- *Complex Geometry File, CGF*.

Statistics files are generated either by PBM Drawer or Interactive Drawer which is explained in Chapter 5.

Simple Geometry File (SGF) Organization

SGF file contains only line geometry features and internal organization of the file and its fields are as follows;

- Group ID: If the network is divided into subgroups which are representing a common property, then group id is assigned an integer value. For example, roads of the same district.
- Feature ID: Feature id is used to represent the feature uniquely within the whole network.
- Feature Type: Represents the type of the feature. Three different geometries are supported by PBM System; line, arc by three coordinates and cubic spline, but SGF file contains only line geometry so feature type is equal to 'LINE'.
- Point Count: Number of points that construct this geometry. For SGF it is 2.
- Point1 ID: Line is defined by using two coordinates, first one is named as Point1 and other is Point2. Here Id of start node or Point1 is given.
- Point1 X: X directional component of the coordinate representing the line start.
- Point1 Y: Y directional component of the coordinate representing the line start.
- Reference Node IDs for Point1: If this coordinate is shared by another line(s) then id representing the start or end node of that line is stored as referenced node id in this field.
- Point1 Mean X: Mean calculated from the sample files which represents the X directional component of the coordinate.
- Point1 Mean Y: Mean calculated from the sample files which represents the Y directional component of the coordinate.
- Point1 Variance X: Variance calculated from the sample files which represents the X directional component of the coordinate.
- Point1 Variance Y: Variance calculated from the sample files which represents the Y directional component of the coordinate.

- Point2 ID: Id of end node or Point2 is given.
- Point2 X: X directional component of the coordinate representing the end of line.
- Point2 Y: Y directional component of the coordinate representing the end of line.
- Reference Node IDs for Point2: If this coordinate is shared by another line(s) then id representing the start or end node of that line is stored as referenced node id in this field.
- Point2 Mean X: Mean calculated from the sample files which represents the X directional component of the coordinate.
- Point2 Mean Y: Mean calculated from the sample files which represents the Y directional component of the coordinate.
- Point2 Variance X: Variance calculated from the sample files which represents the X directional component of the coordinate.
- Point2 Variance Y: Variance calculated from the sample files which represents the Y directional component of the coordinate.
- Variance-covariance matrix members are given separately for each band model supported by PBM System. These are respectively;
  - G-Band Model
  - Line Based Propagated Error Band Model
  - Network Based Propagated Error Band Model

All these fields are written to a file considering each line separately. A sample SGF file content can be viewed in Appendix A.2. SGF files are having an extension of '*.SGF'.

SGF file reader handles reading operation and represents the file content as a collection of Java objects which implement the interface *ICGFFileObject*. This interface is implemented by SGF and CGF file objects, *SGFLineFeature*, *CGFLineFeature*, *CGFArcFeature*, and *CGFSplineFeature*. Hence, each of them present nineteen methods which

exist in the interface. Eighteen of these methods include getter and setter methods for attributes which can also be viewed in Appendix B.2;

- Feature Type is an integer value which is defined to represent the geometry. Possible selections are defined as static final attributes within the *ICGFFileObject* class by considering the supported geometries. For line geometry it is 0; for arc by three coordinates, it is 1 and finally for cubic spline it is defined as 2.
- Array of means for coordinate pair X
- Array of means for coordinate pair Y
- Array of variances for coordinate pair X
- Array of variances for coordinate pair Y
- Array of standard deviations for coordinate pair X
- Array of standard deviations for coordinate pair Y
- Confidence value which controls the width of constructed band
- Array of CovarianceHolder (see Kurtar (2006)) instances used to hold the values of variance covariance matrix

And last method is;

- **GKLinearRing getConstructedGBand():** which returns constructed G-band geometry as GeoKIT class *GKLinearRing*.

SGFFileReader class is constructed by a directory information and selected file name as seen in the code block given in Appendix B.3. Then *SGFFileReader*'s *readFileContent()* method connect to given directory path, read the selected file. Source code of the method can be seen in Appendix B.3. Finally, *SGFFileReader* class contains a vector of *SGFLineFeature* objects which are implementing the interface *ICGFFileObject*.

Complex Geometry File (CGF) Organization

CGF file contains line, arc by three coordinates and cubic spline geometry features and internal organization of the file and its fields for each geometry type is explained

below. Although attributes within the file are common for every supported geometry, number of occurences are varying according to number of points defining the geometry. For example; for a line feature coordinate based variables are repeating 2 times, for arc 3 times and for cubic spline 4 times. Common fields presented in the file organization are;

- Feature ID: Feature id is used to represent the feature uniquely within the whole network.
- Feature Type: Represents the type of the feature. Three different geometries are supported by PBM System; line, arc by three coordinates and cubic spline. Feature type indicators are 'LINE', 'ARC' and 'SPLINE', respectively.
- Point Count: Number of points which construct the geometry. For line features this value is set to 2, for arc 3 and for spline 4.
- Point ID: Id of the geometries' control point
- Point X: X directional component of geometries' control point
- Point Y: Y directional component of geometries' control point
- Point Mean X: Calculated mean value for the X directional component of geometries' control point
- Point Mean Y: Calculated mean value for the Y directional component of geometries' control point
- Point Variance X: Calculated variance value for the X directional component of geometries' control point
- Point Variance Y: Calculated variance value for the Y directional component of geometries' control point
- Variance-covariance matrix members are given separately for G-band model supported by PBM System.

All these fields are written to a file considering each geometry separately. Fields defined above are repeated according to the number of control points. A sample CGF

file content can be viewed in Appendix A.3. CGF files are having an extension of ' *.CGF '.

CGF file reader class *CGFFileReader* handles reading operation and represents the file content as a collection of Java objects which implement the interface *ICGFFileObject*. This interface is implemented by CGF file objects, *CGFLineFeature*, *CGFArcFeature*, and *CGFSplineFeature*. Hence, each of them present nineteen methods defined in CGF file organization part. See Appendix B.2.

CGFFileReader class is also constructed by a directory information and selected file name. Then *CGFFileReader*'s *readFileContent()* method connect to given directory path, read the selected file. See Appendix B.4 for Java code. Finally, *CGFFileReader* class contains a vector of *CGFLineFeature*, *CGFArcFeature*, and *CGFSplineFeature* objects which are implementing the interface *ICGFFileObject*.

4.4.2 Band Generation

PBM Band Generation Software is designed as a wizard to guide the user step by step (See Figure 4.6). Process workflow diagram of network band generation software is visualized in Figure 4.7.

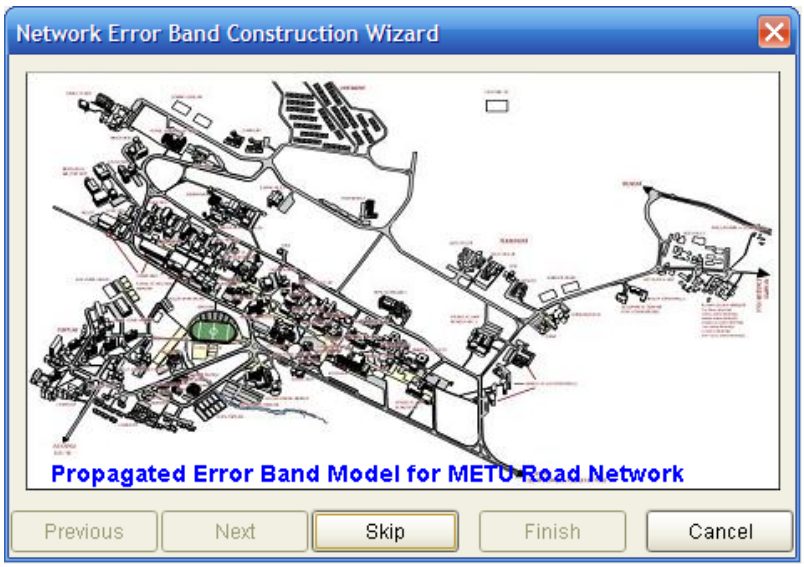


Figure 4.6: PBM Band Generation Wizard

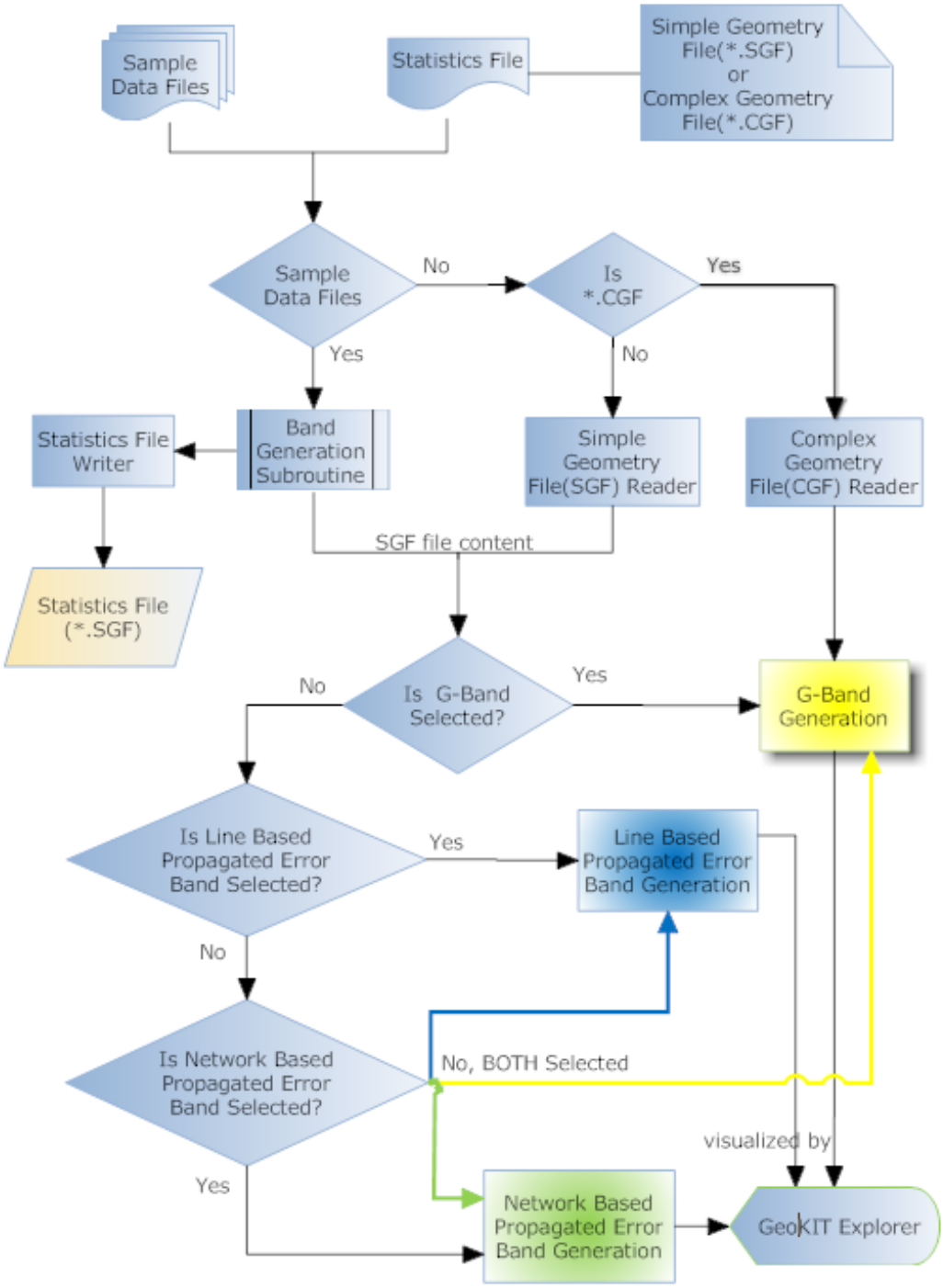


Figure 4.7: PBM Software System Flowchart

As seen in workflow diagram of the system (Figure 4.7), band generation process is handled in two different ways. In the first one, band generation process runs over sample data created by Sample Collector Applet (Section 4.2). In the second one, it is done by using selected statistics file. Process flow and wizard pages are broadly explained in the following sections.

4.4.2.1 Band Generation from Sample Data

Band generation process starts with selecting the input type. If user selects "*Construct network file from samples*" option as seen in Figure 4.8, band generation process is handled using the sample files which exist under the given directory path. User has to select the directory of sample files and press next button to reach latter wizard page. At the moment of pressing the next button of the wizard, PBM Band Generation Software's responsible class, named **PBMStartingPage**, passes selected directory path and selected input operation type to next wizard page, **PBMFileSelectionPage**. This transfer is done via **public WizardGraphPage getNext()** function of **PBMStartingPage**. In this function, next wizard page is constructed and necessary parameters are passed. This new page is visualized when "**Next**" is pressed.

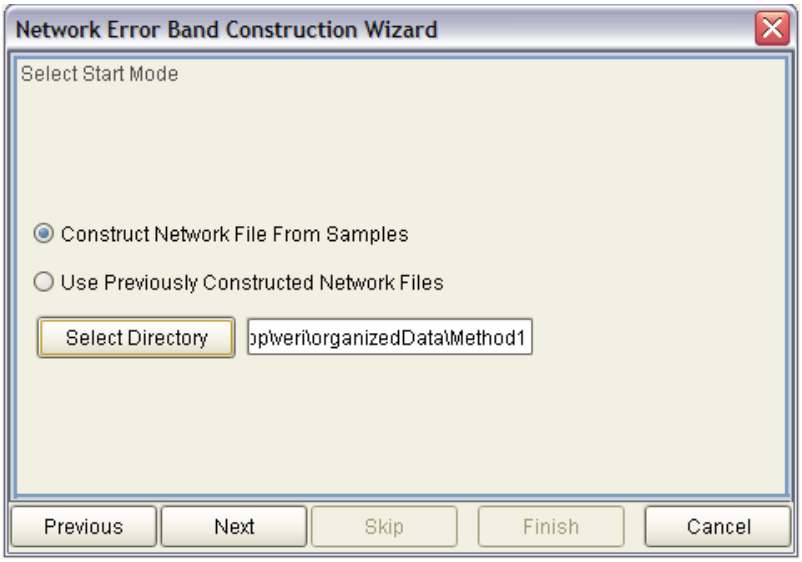


Figure 4.8: Input Type Selection Wizard Page - Starting with Sample Files

PBMFileSelectionPage connects to the specified directory and gets all files with extension of "*.txt". After that, it initializes the GUI(Graphical User Interface) and

displays name of sample files in a list as seen in Figure 4.9. User has to select some sample files from the "Available Sample Files" list for band generation process and move them to "Selected Sample Files" list. Minimum number of selection is set to 5 and if number of selected sample files is less than 5, wizard would not allow to continue. If user wants to create a statistics file from the sample inputs, he/she has to change the state of "Create Statistics File" to ON, and then define a file name and a path. Statistics file is created during band generation process, and saved to the specified directory with the given name. After all, user can press "**Next**" button to proceed.

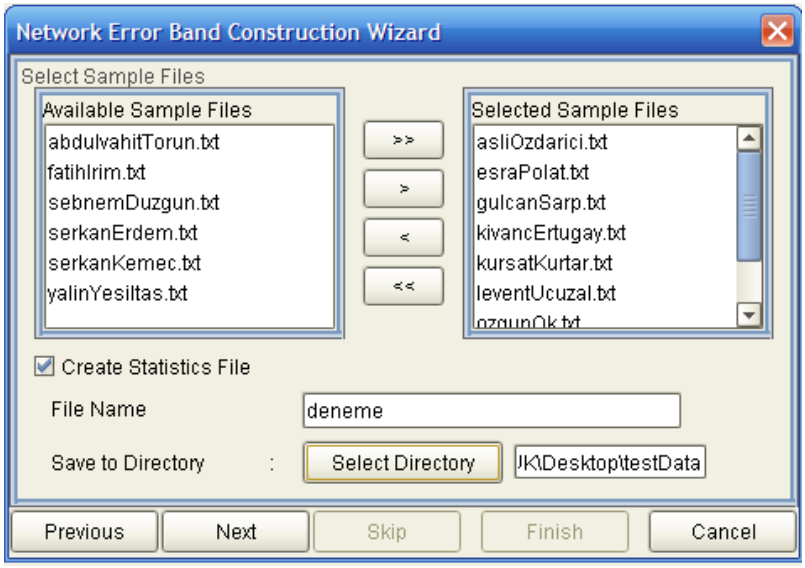


Figure 4.9: Sample File Selection Wizard Page

Incoming page is **PBMMethodSelectionPage** which can be viewed in Figure 4.10. User can change confidence level and select one of the methods listed in the wizard page;

- G-Band Model,
- Line Based Propagated Error Band Model,
- Network Based Propagated Error Band Model, and
- Both Defined Models.

After selecting the band type, "Next" button has to be pressed to proceed. This action activates sample data reading process and calculates the necessary statistics which are used during band generation. SGF file content explained in Section 4.4.1.2, contains these statistics which are derived during band generation process. Band generation model which is implemented in the application, is explained in Chapter 2.

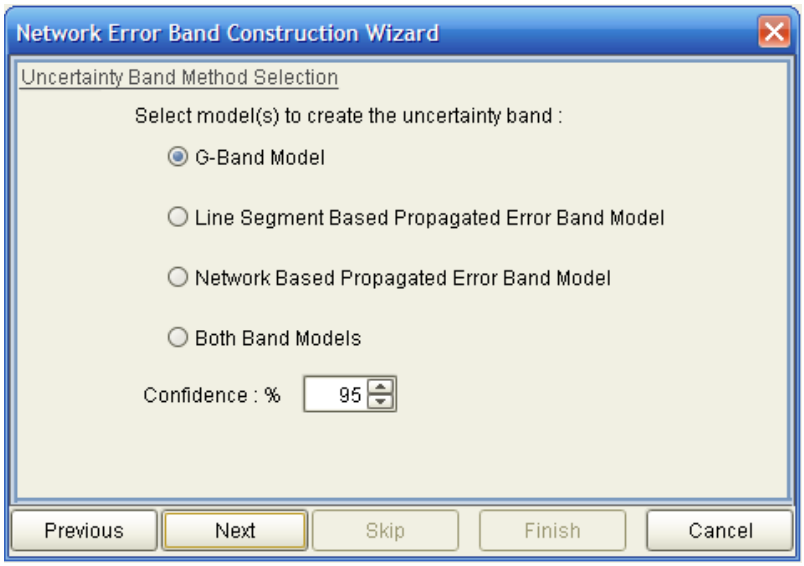


Figure 4.10: Uncertainty Band Method Selection Wizard Page

Sample data reading process is handled by **PBMSampleFileReader** class which is in package **metu.ggit.thesis.pbm.reader.samplefile**. This class is constructed by selected directory path info and a collection of selected sample file names. Within the constructor **void readSampleFiles()** method is called and at the end of reading, sample files' content is represented as a collection of **PBMSampleLine** class instances. Remaining parts of the procedure use this collection. **SGFFileWriter** controls the flow of data export, calculation of band statistics and creation of SGF statistics file. **SGFFileWriter** initializes **PBMBandStatisticsCalculator** class and handle each line in a routine which calls

- **PBMNetworkLine calculateStatistics(GKCoordinateArray arrayStartNode, GKCoordinateArray arrayEndNode, int index, PBMSampleLine originalLine, boolean baseGiven)**

method of **PBMBandStatisticsCalculator**. See Appendix B.5 for java code of specified function. This method accepts coordinate array of start and end points, index of processed line, original line information and a boolean variable representing the existence of true values (assumed to be true). This variable is important because if true values are not provided then calculated means from sample files are used instead of true value. Generated **PBMNetworkLine** class instance is returned back to the **SGFFileWriter**. In the same manner, for line based propagated error band model, **PBMBandStatisticsCalculator**'s

- **void calculatePropagatedStats4Line(PBMNetworkLine nLine, GKCoordinateArray arrayAngleLength, GKCoordinateArray arrayNewEnd)**

method is called. See Appendix B.5 for java code of specified function. This method accepts the generated **PBMNetworkLine** instance, array of calculated angle and length values for each line, array of generated new end points using the angle and length values of original data to reveal the error between sample coordinates and original coordinates.

Similar calculation is handled for network based propagated error band model and **PBMBandStatisticsCalculator**'s

- **void calculatePropagatedNetworkStats(PBMNetworkLine previousLine, PBMNetworkLine currentLine)**

method is called for each line during the process. See Appendix B.5 for java code of specified function. This method gets two **PBMNetworkLine** objects. Second parameter of function is the currently processed line segment. First parameter of the function represents the previous line which is strictly connected to the current line segment. Current line's internal error propagation calculation is similar to the method explained in the previous paragraph. But, this time propagated error which is present at the end point of the previous line is used. And finally statictics and band generation is done in function;

- **void calculatePropagatedStats4Network(PBMNetworkLine nLine, GKCoordinateArray arrayAngleLength, GKCoordinateArray arrayAngle, GKCoordinateArray arrayLength)**

of **PBMBandStatisticsCalculator**. See Appendix B.5 for java code of specified function.

New coming page is **PBMParameterSelectionPage** and included parameters are analyzed in three subgroups. These groups can be seen in Figures 4.11, 4.12 and 4.13. In the first group, parameters which are related with true and sample geometries are given. In the second group, band geometry related parameters are served. Finally in the third group, comparison parameters are analyzed. Activation of last group is controlled by "Both Band Models" representation given in Figure 4.10, otherwise this tab may not be visualized.

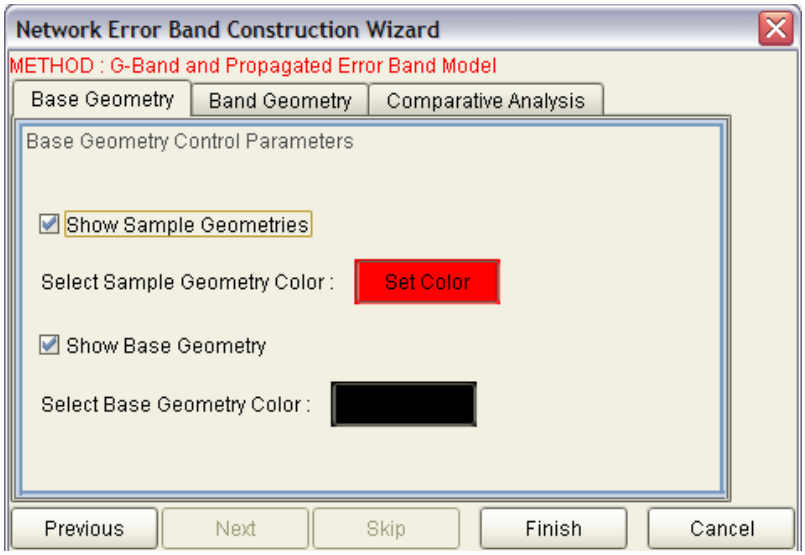


Figure 4.11: PBM Base Geometry Parameters Selection Wizard Page

Visualization of true and measured data for selected network can be controlled by the help of wizard page in Figure 4.11. If band generation process starts by statistics file, control parameter "Show Sample Geometries" and its color setting option is removed from the GUI content, otherwise it is completely same as Figure 4.11.

Uncertainty band parameters are controlled by the help of user interface seen in Figure 4.12. User can either edit display color of band or can turn display option on for labels used to give extra information about control points of lines and statistics calculated for that points. "Apply Overlay(UNION)" option applies a union operation for created band geometries and displays a single polygon for the whole network.

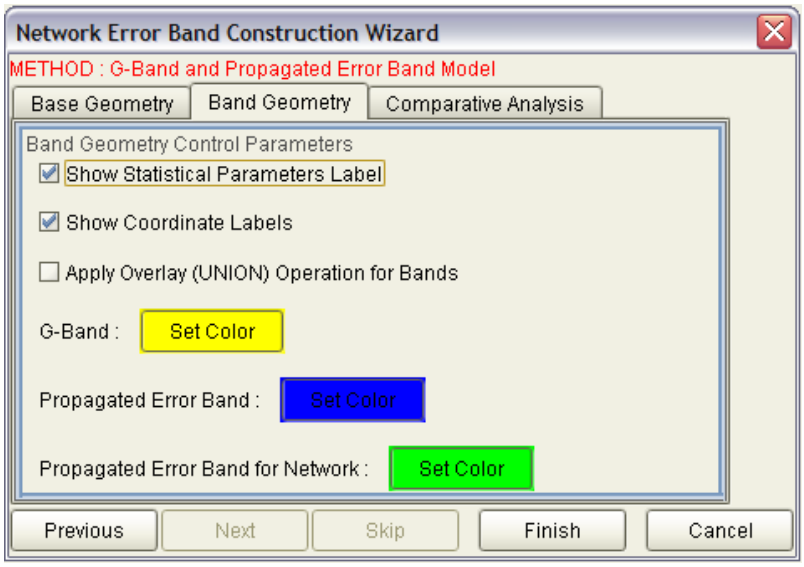


Figure 4.12: PBM Band Geometry Parameters Selection Wizard Page

As defined in the previous paragraphs, comparison parameters' control panel is visible, if user selects the method of applying both band models because these parameters are used during the comparison of G-band and propagated error band model. Provided comparison parameter types are; areal comparison, sample geometry coverage and intersections. Comparison control parameters can be seen in Figure 4.13.

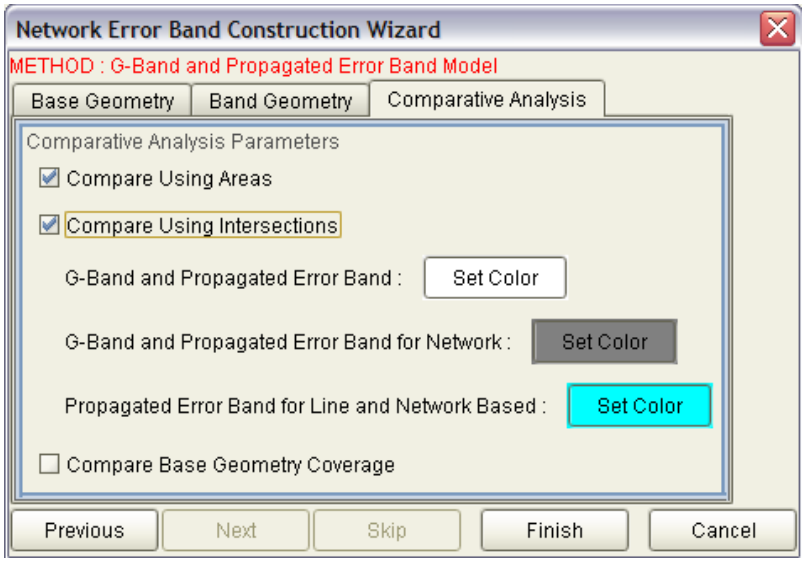


Figure 4.13: PBM Comparison Parameters Selection Wizard Page

Areal comparison parameter diplays all generated band areas in a table considering each line separately. It also visualizes the total which is generated by each method. The comments on results take place in Chapter 6.

Sample geometry coverage gives number of lines that lie inside and outside of the generated uncertainty band. Tables D.4, D.5 and D.6 show the outcomes of the sofware. These results are also commented in Chapter 6.

Last comparison parameter is visualization of intersections. Applied band models listed in Figure 4.10 are coupled and for each pair intersection and difference operations are applied. And finally the results are visualized by GeoKIT. These outcomes are also commented in Chapter 6.

After all, user can press "Finish" button to finalize the wizard. At that moment all control parameters are passed to **PBMInteractiveLayer** class which is responsible of rendering. Rendering is control by using the predefined color and display parameters. Finally results of band generation process are diplayed in GeoKIT Explorer's mapview window and created legends and tables are also displayed at the bottom tab of GeoKIT.

4.4.2.2 Band Generation from Statistics File

Band generation using previously created statistics file is the other way to enter PBM Band Generation System. File organization (in Section 4.4.1.2) and process flow (Figure 4.7) are explained in previous sections.

User can start band generation process by selecting option ' Use Previously Constructed Network Files ' and defining a suitable directory path for existing files with extension of "*.SGF" or "*.CGF". When "Next" button of user interface given in Figure 4.14 is pressed, files with specified extensions are extracted and passed to the constructor of the class **PBMStatFileSelectionPage** by **PBMStartingPage**.

PBMStatFileSelectionPage initializes the user interface which can be seen in Figure 4.15. User has to select one of these listed files and press "Next" button to proceed. At that moment, selected file content is parsed and represented by Java objects. If file with an extension of "*.SGF" is selected, reading operation is handled by **SGFFileReader**(see Appendix B.3), otherwise it is handled by class **CGFFileReader**(see Appendix B.4). Source code of the project can be found in pack-

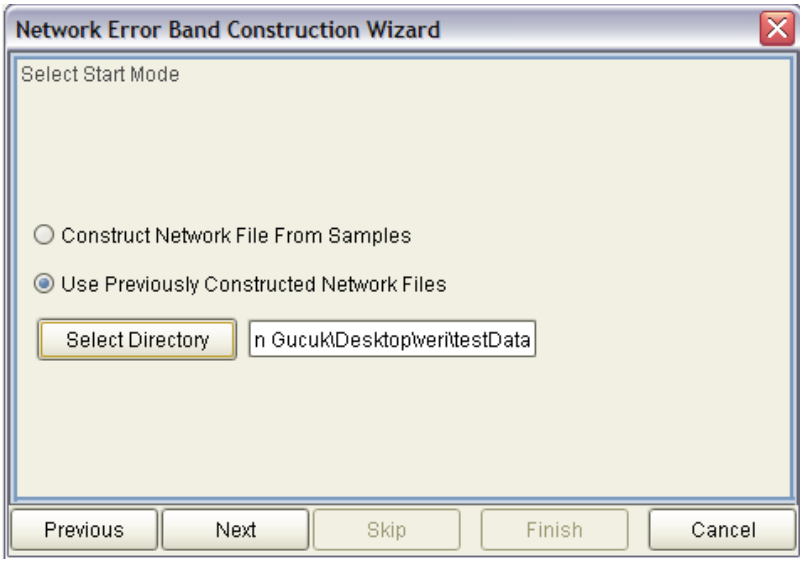


Figure 4.14: Input Type Selection Wizard Page - Starting with Statistics File

ages "**metu.ggit.thesis.pbm.reader.sgf**" and "**metu.ggit.thesis.pbm.reader.cgf**". After parsing the selected file, vector is constructed from **SGFLineFeature**, **CGFLineFeature**, **CGFArcFeature** and **CGFSplineFeature** instances by the specified file parser. These classes implements the interface **ICGFFileObject**. Hence, each of them presents nineteen methods defined in SGF file organization part of this thesis.

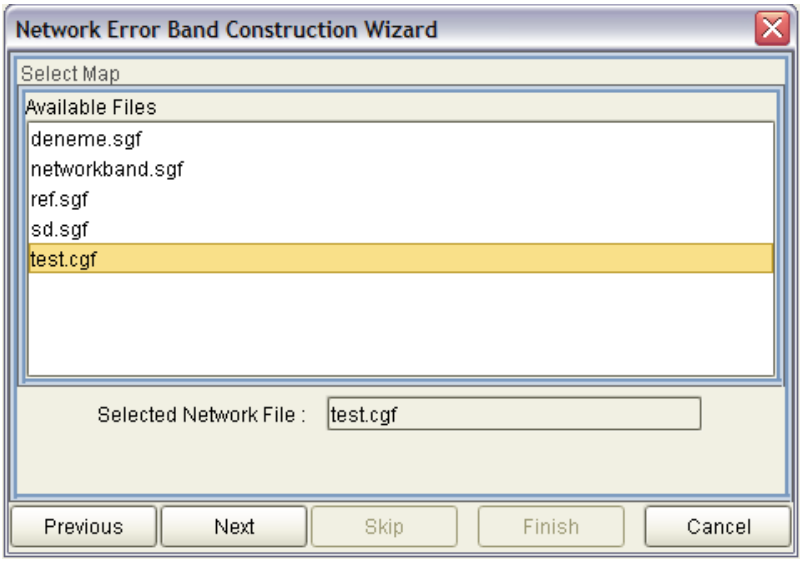


Figure 4.15: Statistics File Selection Wizard Page

Following wizard page changes according to the performed selection type. If

CGF statistics file is selected, state of the process is directed to parameter selection wizard page seen in Figure 4.11 because for CGF objects, only the proposed G-band model introduced by Kurtar (2006) is used. If SGF file is selected, next state of the wizard is method selection page as seen in Figure 4.10. Following states are same as the ones explained in band construction using sample files (Section 4.4.2.1).

4.5 Technical Specifications

The PBM System module and its application are created by Java using commercial GIS API, GeoKIT 2.6. Developed applications can easily be plugged-in to GeoKIT Explorer. Platform independency provided by Java is admitting the run of the software at any platform.

The performance of the system is mostly affected from the complexity of the selected line network and used geometries. For this reason for same line network, computation time needed for geometries introduced in CGF content requires higher rates than line geometry introduced in SGF content.

CHAPTER 5

INTERACTIVE NETWORK DESIGN SOFTWARE

Interactive **D**rawer (ID) plug-in application is used by PBM System as a source which generates simulated data. This application lets the user to define a network which is composed of either linear or curvilinear geometric features including line, arc by three points and cubic spline. Figure 4.2 visualizes the interaction of the application both with the user and the PBM system architecture. Any supported geometric feature can be defined by user for a given standard deviation. Specified data is simulated using Monte Carlo method and then generated network statistics are written to a file which is later used by PBM Drawer during band generation. ID application is also coded as a plug-in to GeoKIT Explorer. Explanation of system work flow, generated user interfaces and process organization of ID plug-in are the subjects mentioned in this chapter.

5.1 Interactive Drawer Plugin Application

Interactive drawer work flow diagram can be seen in Figure 5.1. As seen from the figure, four main actions are controlling the process flow. These are;

- Activate Handler Action
- Clear All Action
- Approve Geometries Action
- Save Work Action

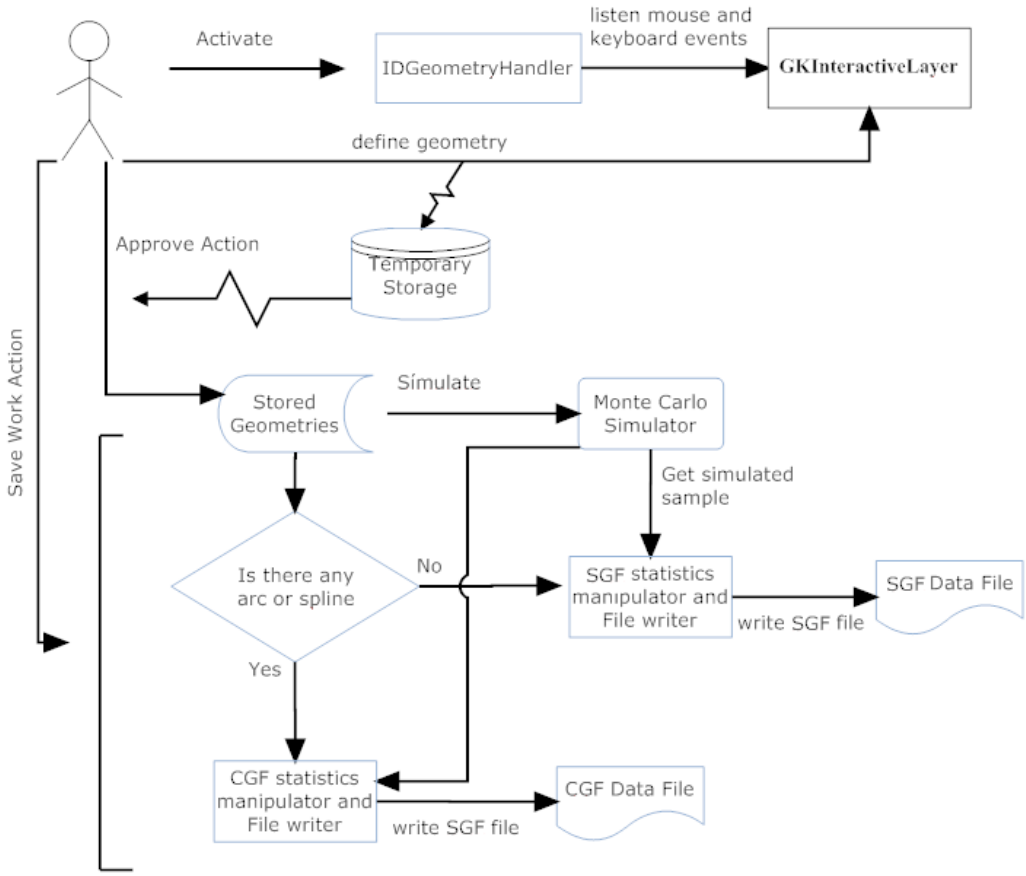


Figure 5.1: Interactive Drawer Work Flow Diagram

Activate Handler action activates the specified geometry handler which is explained in Section 5.1.3. This handler is activated by making a selection for the needed geometry. Handler activation process and its interaction with the user interface are explained in Section 5.1.1. After the activation of geometry handler, user can define geometries using mouse actions i.e. Specify points over the map view of GeoKIT Explorer.

Clear All action cleans all geometries, which exist in the temporary geometry storage component of the previously defined handler action. An important point to be considered here is that *Clear All* action deletes only **not approved** geometries, which are on the fly .

Approve action accepts geometries which are defined up to that time, into a stable storage area. From now on these approved geometries can not be removed from the system unless the whole application is closed. *Approve* action involves simulation process inside and this mechanism is broadly expressed under Section 5.1.4.

Finally the *Save Work* action calculates the necessary statistics for approved geometries and writes either a SGF or a CGF file by considering the defined geometry types. Internal structure and organization of statistics files are explained in Chapter 4 under the titles of ' Simple Geometry File (SGF) Organization ' and ' Complex Geometry File (CGF) Organization ' in Section 4.4.1.2.

Following sections focus on the specified actions considering the GUI connections and applied methods.

5.1.1 ID Graphical User Interface (GUI)

General overview of the developed user interface for ID application can be seen in Figure 5.2. ID application GUI is composed of three main panels which are;

1. Action Controller Buttons Panel, located at the top,
2. Geometry Control Panel, located on the left,
3. Statistical Calculations Control Panel, located on the right.

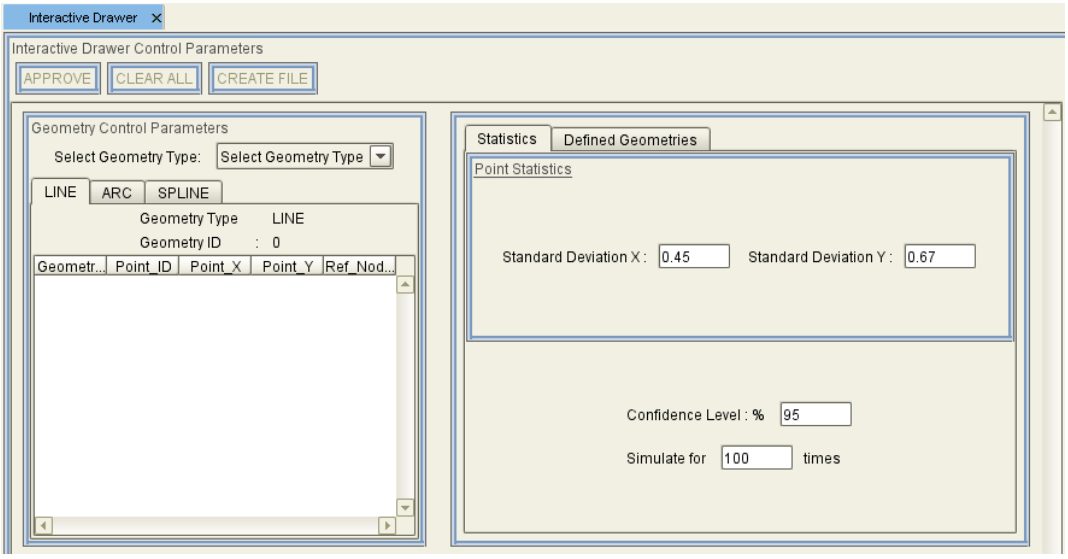


Figure 5.2: Interactive Drawer General Overview

Geometry Control Panel visualizes the defined geometries separately for each supported feature type. User defined geometries are listed under the related feature type list. Created lists for line, arc and spline geometries can be viewed in Figures 5.3, 5.4 and 5.5, respectively. Defined geometries are represented based on points constructing the geometry. For this reason, in line list seen in Figure 5.3, a line segment covers two rows of the list. Geometry id given in the first column of the list helps to decompose the content of the list. Same condition is available for arc and spline geometries. This time for three rows within the list define a single arc, similarly four rows define a single spline feature.

Statistical Calculations Control Panel visualizes the initially set statistical variables and control parameters for Monte Carlo simulation method. User can edit any of these parameters. *Defined Geometries* tab of the panel is used to display **APPROVED** geometries included with the results of Monte Carlo Simulation method. Details are given in Section 5.1.4.

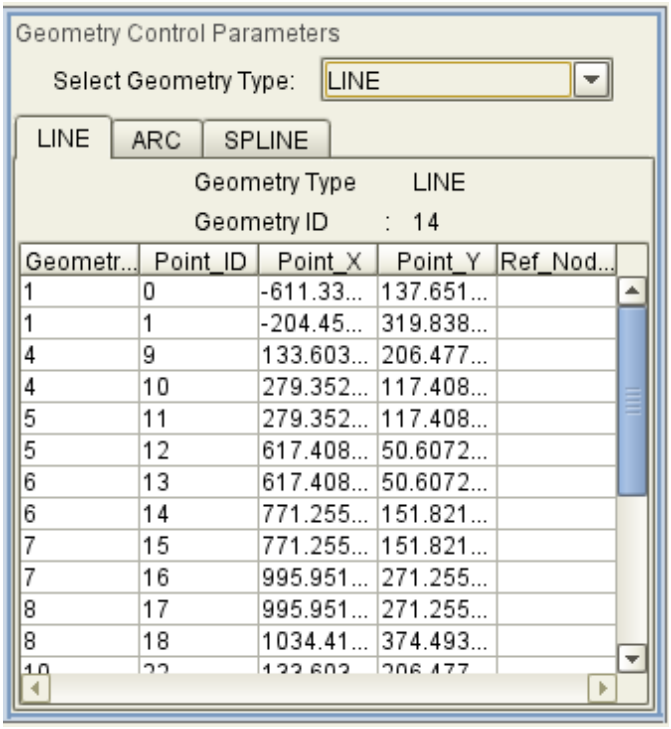


Figure 5.3: Interactive Drawer Line Geometry Definition Panel

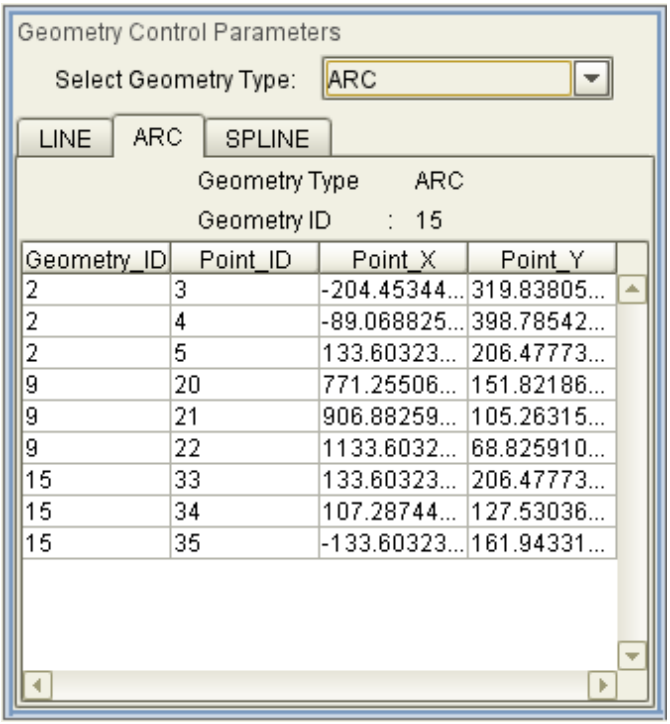


Figure 5.4: Interactive Drawer Arc By Three Coordinates Geometry Definition Panel

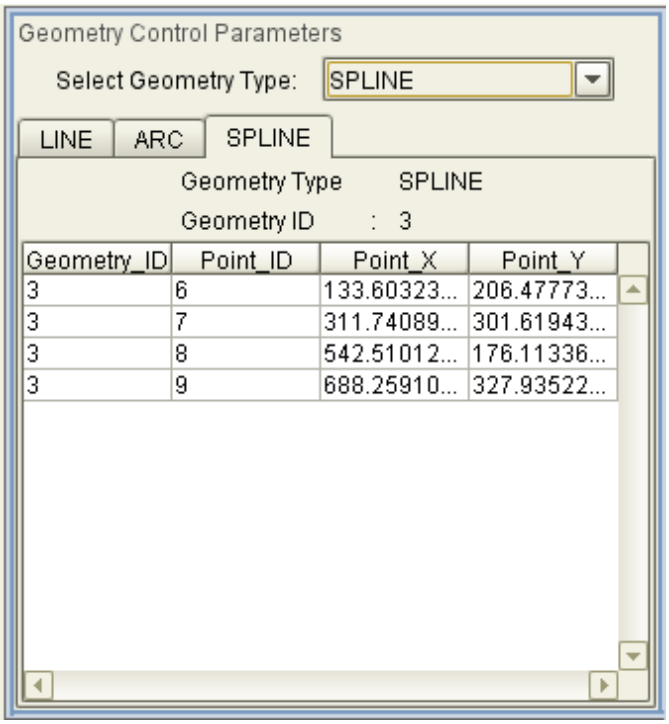


Figure 5.5: Interactive Drawer Cubic Spline Geometry Definition Panel

5.1.2 Supported Geometries

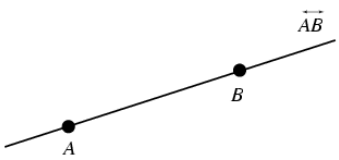
ID application supports three geometries;

- Line Segment
- Arc By Three Coordinates
- Cubic Spline

A line is a straight one-dimensional extending infinitely in both directions (Figure 5.6(a)). A line is uniquely determined by two points, and the line passing through points A and B is denoted $\overleftrightarrow{AB}$. Similarly, the finite line segment terminating at these points is denoted $\overline{AB}$ (Figure 5.6(b)). Line segment is defined as a closed interval corresponding to a finite portion of an infinite line (WolframMathWorld_Line, 2007).

Arc is any smooth curve joining two points. Arc is any portion of the circumference of a circle (Figure 5.7).

Arc is a functional curve and arc by three coordinates is a type of arc which is contructed from three coordinates, seen in Figure 5.8. Types of arc and functions which create an arc can be found in Kurtar (2006).



(a) Line (Adapted from WolframMathWorld_Line (2007))

(b) Line Segment (Adapted from WolframMathWorld_Line (2007))

Figure 5.6: Difference between (a) Line, and (b) Line Segment

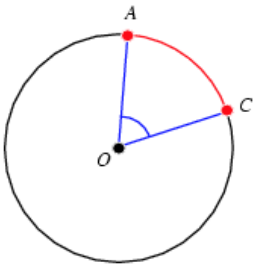


Figure 5.7: Arc as a portion of circle circumference

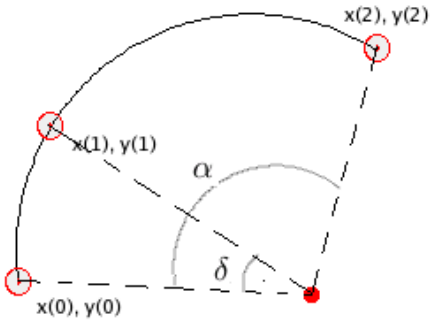


Figure 5.8: Arc by three coordinates (Adapted from Kurtar (2006))

Cubic splines are a sequence of segments each with its own defining function. A cubic spline uses four control points and these four control points are used to solve a 3rd degree polynomial interpolation (Figure 5.9). 3rd degree polynomial interpolation formula can be found in Kurtar (2006).

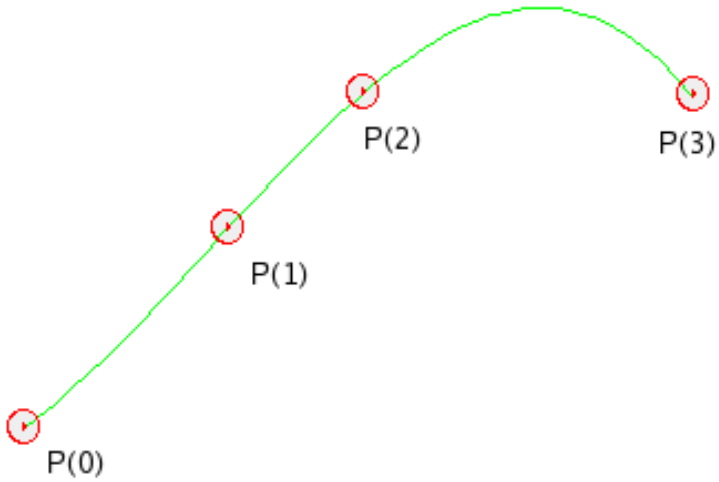


Figure 5.9: Cubic spline (Adapted from Kurtar (2006))

5.1.3 ID Geometry Handler

Drawing utility of the application uses interactive layer properties and functionality of GeoKIT API. **GKInteractiveLayer** is a base class for layers that needs user interaction and it inherits necessary methods from interfaces to control keyboard and mouse events in addition to view specific methods from implemented handlers.

IDGeometryHandler class of the ID software extends *GKInteractiveLayer* to capture the events of the user who uses the software.

public class *IDGeometryHandler* **extends** GKInteractiveLayer **implements** GKIMakeUpLayer, GKIViewHandler

IDGeometryHandler is constructed when the GUI is initialized but for *IDGeometryHandler* to capture mouse events, user has to select the interested geometry type from the combo box seen in Figure 5.10.

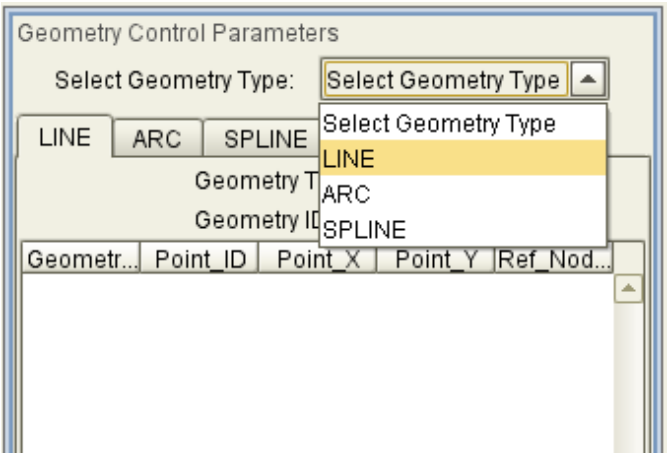


Figure 5.10: Supported geometry type selection combo box

After selecting the feature type, user has to define control points for the selected geometry over the mapview screen of GeoKIT Explorer. Figure 5.11 shows the visualization of three supported geometries in GeoKIT Explorer's map view window.

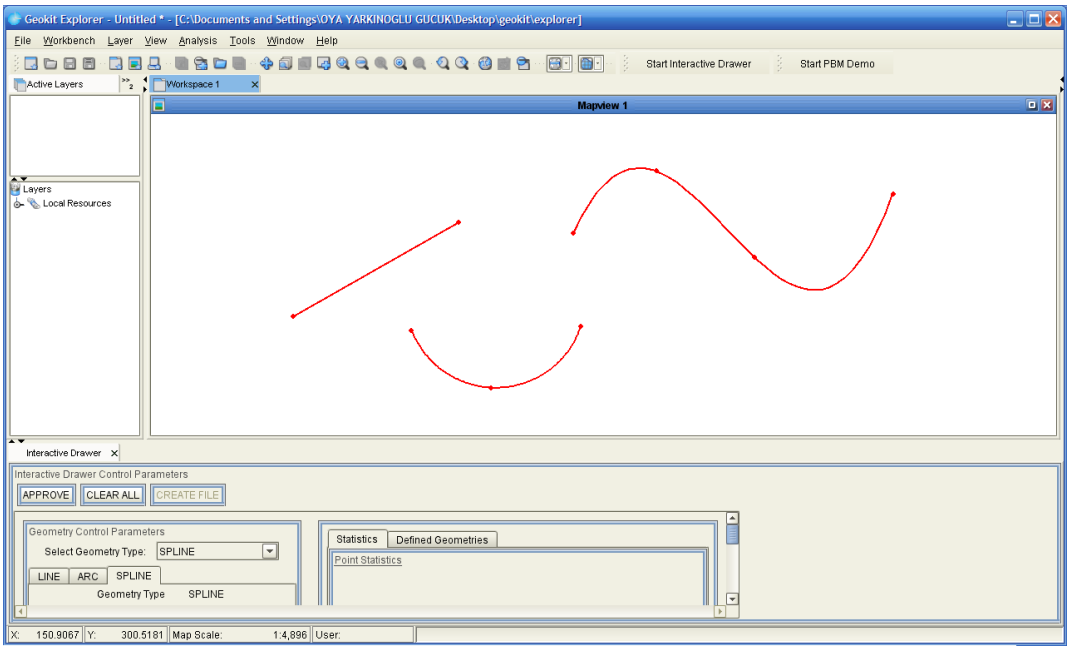


Figure 5.11: Defining line, arc and spline geometries on GeoKIT MapView Screen

IDGeometryHandler matches pre-located points of the defined geometries. This utility is important to provide the *continuity* of the network. In Figure 5.12 it can be seen that matched coordinate is displayed as a large green point. Handler displays

this green point when the mouse comes two pixels near to a predefined geometry control points. This process mainly listens all mouse events during the drawing process and runs over the vector of predefined geometries to find a matching one.

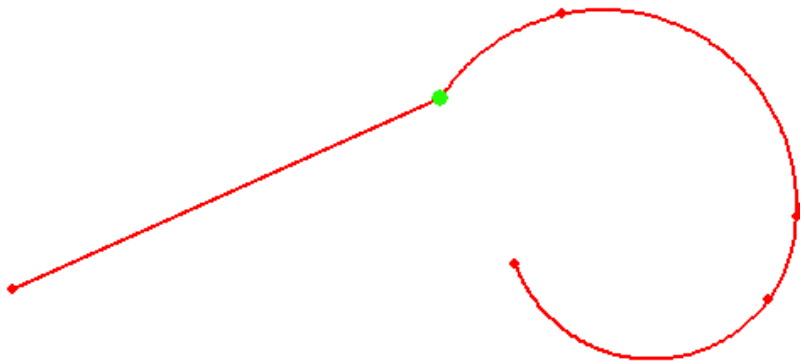


Figure 5.12: Matching point in geometry connections to ensure continuity

User can control geometry drawing process by the help of a pop-up menu which appears over the map view frame when mouse is right clicked. Figure 5.13 represents the popup items. User can clear last geometry or all defined geometries. *Finish Geometry* item finalizes the construction of geometry.

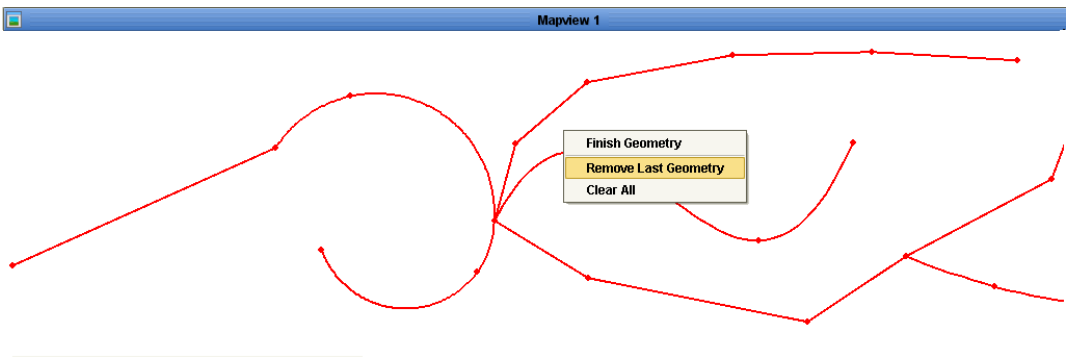


Figure 5.13: Controlling drawing process via popup menu

Finally, when the user finishes defining his network, *Approve* button has to be pressed to change the state and also the storage place of the features. Related code block for *Approve* action can be found in Appendix C.1. During *Approve* action pro-

cess, Monte Carlo simulation method is used to generate simulated sample data for the user defined network structure. Following section focuses on this method and its application through the software.

5.1.4 Monte Carlo Simulator

As mentioned in Chapter 1, Choi (2005) classifies probabilistic methods as statistical and non-statistical methods. Monte Carlo method is a classical method which is widely used in GIS during the analysis of uncertainty. Choi (2005) named Monte Carlo Methods as "statistical simulation methods". Monte Carlo method procedure involves:

1. Generating a set of values by random sampling the known or assumed probability density function for each input variable,

2. Executing an experiment and collecting the data for each of the generated samples, and

3. Employing statistics for the output data set to define its probability density function.

In thesis application, Monte Carlo is used to generate a sample space for given probability distributions. Code block which handles the procedure can be viewed in Appendix C.4. **PBMMonteCarloSimulator** is constructed during *Approve* action (Appendix C.1) by parameters; standard deviation for both directional pairs of coordinate and simulation count.

...

simulator = new **PBMMonteCarloSimulator**(stdevX, stdevY, simulationCount);

...

Simulation results are then filled into a list separately for geometries. Figures 5.14, 5.15 and 5.16 represent the content of the *Defined Geometries* panel for line, arc and spline geometries respectively.

Up to now, network geometries are constructed and appropriate sample space is created via Monte Carlo simulation method. Now all this work has to be organized and to be written to a file for PBM system to recognize both geometries constructing the network structure and derived statistics needed for band generation. This con-

Statistics | Defined Geometries

Geometry_ID	Geometry_Type
1	LINE
4	LINE
5	LINE
6	LINE
7	LINE
8	LINE
10	LINE
11	LINE
12	LINE
13	LINE
14	LINE
2	ARC
9	ARC
15	ARC
3	SPLINE

Original Geometry

Point1 (x,y) : (502.0242914979756 , 461.53846153846155)

Point2 (x,y) : (718.6234817813763 , 467.61133603238864)

Simulation Num	P1_X	P1_Y	P2_X	P2_Y
16	501.6524418008645	462.5132710259137	720.8548056121141	468.804721039187
17	501.3211109083348	462.7035212778033	720.8412244131355	469.495843160067...
18	501.740102463801	463.873484675476...	719.787070926511	468.4447348976567
19	501.654054431703...	464.911789710055...	719.6679852073821	468.731883254929...
20	501.9186479290352	465.4035313752641	719.8333626806458	469.0472258301675
21	501.880204353246...	465.268936636364...	720.0331804604978	468.7761529886962
22	502.091380995393...	464.6507421338434	720.2144714025438	468.0431839896762
23	502.2820460875579	464.515895792341...	719.4726330463243	467.3794065552529
24	502.1612454665616	464.8105316683365	719.2086411940993	467.1034453718237
25	502.9196708796428	465.7387609381011	719.0355974787565	468.1634249960789
26	503.3546337418791	465.505784081835...	719.3538620522202	468.7174838156401

Figure 5.14: Simulation results for line geometry

Statistics | Defined Geometries

Geometry_ID	Geometry_Type
1	LINE
4	LINE
5	LINE
6	LINE
7	LINE
8	LINE
10	LINE
11	LINE
12	LINE
13	LINE
14	LINE
2	ARC
9	ARC
15	ARC
3	SPLINE

Original Geometry

Point1 (x,y) : (-204.45344129554667 , 319.8380566801619)

Point2 (x,y) : (-89.06882591093131 , 398.78542510121457)

Point3 (x,y) : (133.60323886639662 , 206.47773279352225)

Simulation N...	P1_X	P1_Y	P2_X	P2_Y	P3_X	P3_Y
6	-204.735359...	319.4465974...	-89.8897292...	399.3987601...	134.1676103...	206.1580929...
7	-204.233579...	319.2105385...	-89.0675548...	400.1828029...	133.8840795...	206.0801461...
8	-204.615930...	318.8948017...	-89.7567765...	398.7348633...	133.5647120...	207.4189427...
9	-204.559568...	319.1584229...	-89.0452223...	399.1830855...	133.3500199...	206.6689222...
10	-204.671561...	319.7780228...	-89.4372084...	397.6534049...	134.2167445...	206.3738991...
11	-205.404817...	319.6877468...	-89.0842253...	399.4676477...	133.5011966...	205.1127360...
12	-204.157510...	320.5789754...	-88.9388014...	398.7425066...	132.8908651...	206.8225736...
13	-204.957116...	319.7464653...	-89.0078877...	397.1140904...	133.8924528...	205.2036450...
14	-204.457812...	319.3805854...	-89.1459169...	399.5434122...	133.6413963...	206.0682053...
15	-204.201977...	320.6183961...	-89.0677659...	398.5954588...	133.0012914...	206.8760802...

Figure 5.15: Simulation results for arc geometry

Statistics | Defined Geometries

Geometry_ID	Geometry_Type
1	LINE
4	LINE
5	LINE
6	LINE
7	LINE
8	LINE
10	LINE
11	LINE
12	LINE
13	LINE
14	LINE
2	ARC
9	ARC
15	ARC
3	SPLINE

Original Geometry

Point1 (x,y) : (133.60323886639662 , 206.47773279352225)

Point2 (x,y) : (311.740890688259 , 301.61943319838053)

Point3 (x,y) : (542.5101214574897 , 176.11336032388664)

Point4 (x,y) : (688.2591093117408 , 327.9352226720648)

Simulatio...	P1_X	P1_Y	P2_X	P2_Y	P3_X	P3_Y	P4_X	P4_Y
9	133.1963...	205.9434...	311.9273...	300.9615...	543.0654...	175.4985...	687.7251...	328.5051...
10	133.5061...	207.1222...	311.9498...	301.9340...	543.0991...	175.4747...	688.0533...	328.6010...
11	133.6466...	206.2261...	311.0912...	300.8879...	542.7884...	175.1459...	688.8712...	327.3480...
12	133.6330...	206.6982...	311.6847...	301.2736...	542.3967...	175.3679...	688.2445...	327.6854...
13	133.4733...	205.6270...	312.1786...	301.5106...	542.0447...	176.0785...	688.7543...	326.0319...
14	133.1716...	206.3444...	311.5078...	302.1559...	542.5197...	176.6026...	688.4925...	327.5436...
15	133.0699...	206.1691...	312.5630...	301.9008...	541.8146...	178.0916...	689.3114...	327.1374...
16	133.3765...	205.3830...	311.4326...	301.8928...	542.0837...	174.3806...	687.8902...	327.6219...

Figure 5.16: Simulation results for spline geometry

nection is built by the help of statistics file either SGF or CGF. Next section expresses the statistics file generation procedure.

5.1.5 Statistics File Writer

Generation of SGF or CGF file is accomplished under the action *Save Work*. System decides on the extension of file by considering the types of defined geometries. If network structure contains arc and spline rather than line geometry, file with an extension of "*.CGF" is created and derived statistics are written into it. Otherwise, SGF file is constructed.

Responsible class for statistics file generation process is **IDStatisticsFileGenerator**. This class is informed about the content of the defined geometries by a boolean parameter passed through constructor.

```
...
public IDStatisticsFileGenerator(boolean cgfFileContent)
    this.cgfFileContent = cgfFileContent;
...
```

After deciding the extension of the file, system displays a dialog for user to enter file name and file save path. When user enters and presses the "*Ok*" button of the dialog which can be viewed in Figure 5.17, *IDStatisticsFileGenerator* calls either method $calculateSGFStatistics\_write2File(\ldots)$ or method $calculateCGFStatistics\_write2-File(\ldots)$ according to the decided file extension.

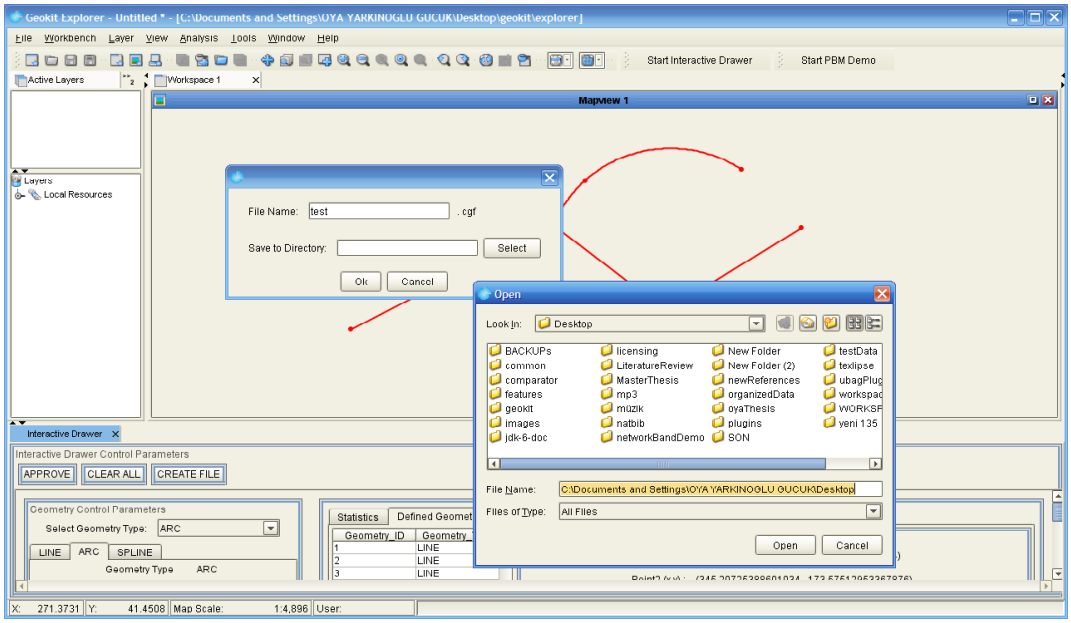


Figure 5.17: Save accomplished work into statistics file

Finally this is the end of interactive drawer's network constructor application. Constructed file(s) can be used to generate band geometries through PBM system.

5.2 Interactive Drawer API Explanation

GeoKIT Core and GeoKIT Explorer modules are used in the implementation of the application and supplies necessary GIS components and GIS functionality to the application. GeoKIT Explorer's Plugable API architecture which serves a high level plug-in API, facilitates the integration of the developed application piece to GeoKIT. Additionally, the practical and friendly user interface of GeoKIT Explorer Desktop Application smarten up the representation of the results.

PBM is another API used during the implementation of the ID application. This API serves uncertainty band generation functionality for line, arc by three coordinates, and cubic spline using U-Bag API to describe a G-Band model explained in Chapter 2. These APIs are developed by using Java and GeoKIT Explorer.

Architectural relation between used and implemented modules can be seen on Figure 5.18. As seen, JVM (Java Virtual Machine) is the base and respectively GeoKIT API, U-Bag API, PBM API and finally the PBM application run over this substructure.

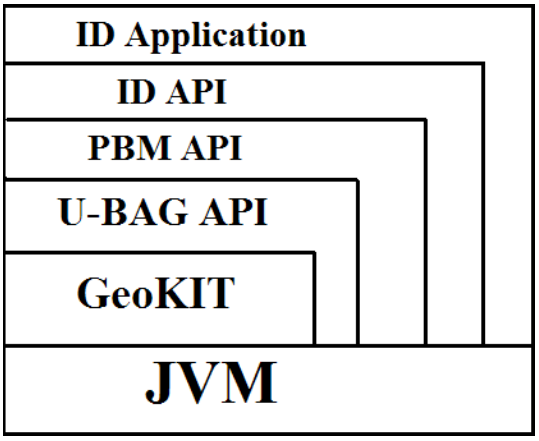


Figure 5.18: System Architecture of Interactive Drawer

5.3 Technical Specifications

The ID System module and its application are created by Java using commercial GIS API, GeoKIT 2.6. Developed applications can easily be plugged-in to GeoKIT Explorer. Platform independency provided by Java is admitting the run of the software at any platform.

The performance of the system is mostly affected from the complexity of the defined network structure, used geometries and selected simulation count values. Computation time needed for generation of the sample space and calculation of the statistics are dependent to this reason.

CHAPTER 6

COMPARISON OF G-BAND AND PROPAGATED ERROR BAND

To explain the pros and cons of the studied propagated uncertainty band models for a linear network and to compare the model to the previously defined G-Band model, this chapter is divided into two parts. In the first part, error measures derived for the sample network data are explained. In the second part, the analysis and the results are compared for implemented uncertainty band models; G-Band Model, Line Based Propagated Error Band Model(LPBM) and Network Based Propagated Error Band Model(NPBM). All these comparative analysis are based on the sample data gained from the users whom digitized the selected line network expressed in Section 3.1.1. Implemented comparative analysis can be grouped into three titles; comparison of band geometry area, comparison of intersection of band geometries and comparison of sample geometry coverage.

6.1 Error Measures of Samples

For the whole network, to compare the variation of sample coordinate population including x and y, coefficient of variations are calculated. For each node, coefficient of variation on x and y are separately calculated and shown in a diagram to visualize the variation on the nodes of the network. The node id versus coefficient of variation on x and y can be seen in Figure 6.1.

It can be easily recognized that variation peaks especially on coordinates with short line length. Network with numbered node ids in Figure 6.2 should be viewed to understand graphical explanations below.

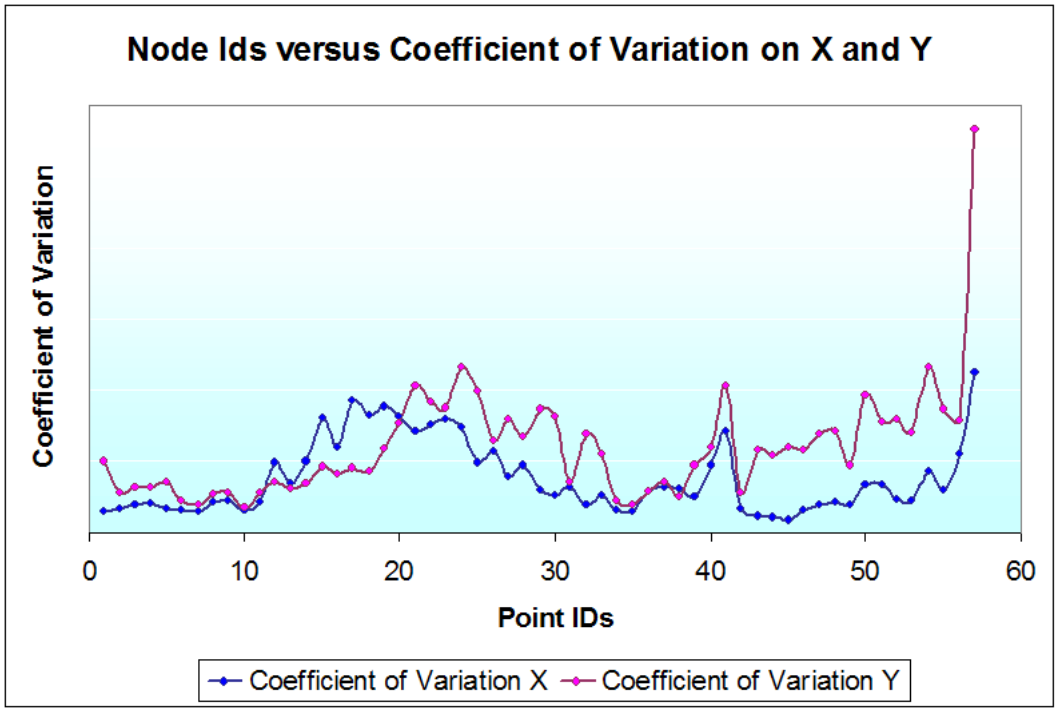


Figure 6.1: Node Id versus Coefficient of Variation on X and Y

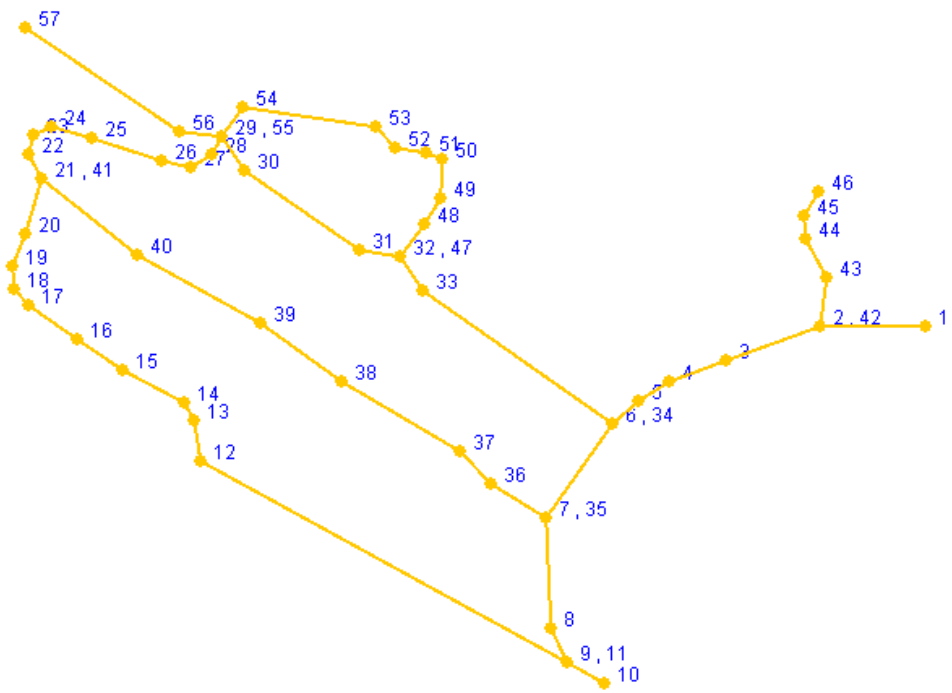


Figure 6.2: Selected Linear Network with Node Ids

In Figure 6.1 coefficient of variation on the coordinates shows an increasing manner starting from node id 10 and makes a peak near node id 20. This increasing manner loses its effect after node id 25 and decreases up to node id 30. If Figure 6.2 is examined to explore the properties of the lines within the given range, it can be noticed that topology of the network is twisted. So the lines that constitute this twisted or curved nature of the topology have short line lengths. These short lines which are constructed by near coordinate pairs, need extra attention during the digitizing process, and consequently they bring about the digitizing error.

Although coefficient of variation around node id 41 seems to make a peak, this is deceptive. As the digitizing software that is used during sample collection process matches the nearest previously entered coordinate value to ensure the continuity of the network; this deceptive peaks occur. These elusive peaks can be distinguished by looking at the conjunctions of the network for shared nodes. For example, the coordinate having node id 41 is also shared by node id 21. Hence increasing coefficient of variation on node 41 is prospective. Another increase at the end of the network in spite of long line length can be explained as the exhaustion and inattentiveness of the people.

Another plug-in application named Sample Comparator is used to verify the results of the graph in Figure 6.1. Variation of each coordinate can be checked over using Sample Comparator Plug-in as seen in Figure 6.3.

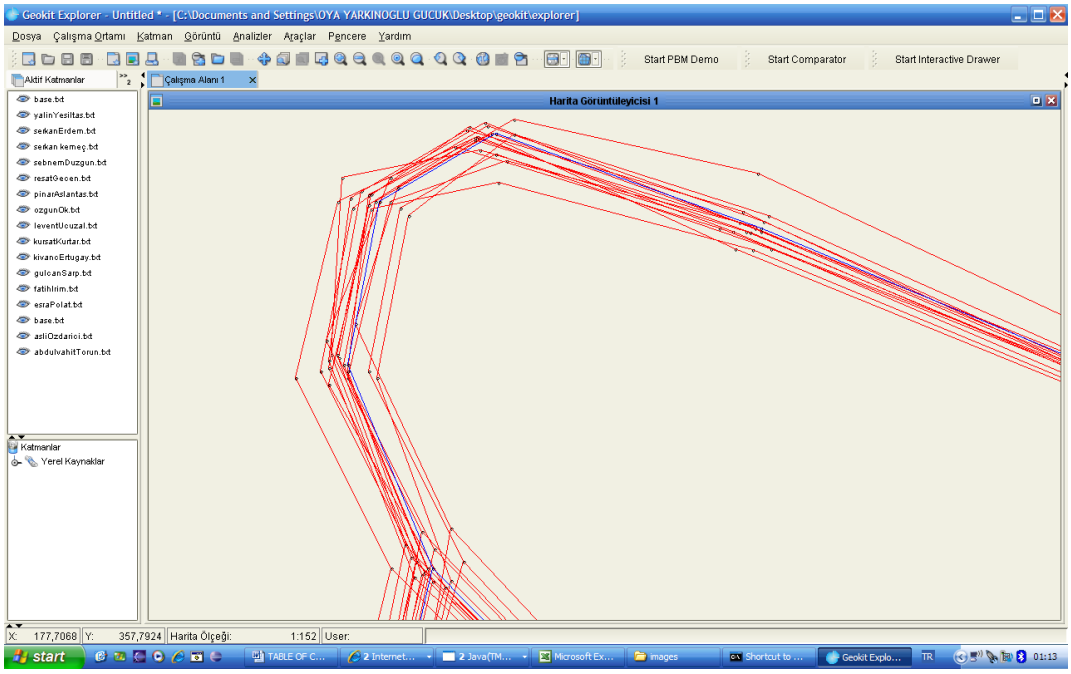


Figure 6.3: Sample Comparator Software Snapshot

6.1.1 G-Band Implementation

G-Band model is introduced by Shi and Liu (2000). The model describes the positional error of line segments in geographical information systems (GIS), like some other models described in Chapter 2. The model is based on a stochastic process theory and models the error of line segment for both independent and correlated cases of the two end points. If stochastic process is directionally independent, G-Band model degenerates to error-band models. If line is homogeneous and directionally independent, G-Band degenerates to a parallel band along line (Shi and Liu, 2000).

If the end points are correlated and dependent on each other, G-Band models a band with larger band width at the end points and it becomes smaller through the inner parts of the line segment. Band width toward the end points are considered separately according to the distribution of the end points. Distribution at the end points can be either equal or unequal to each other. This will effect the shape of the

generated G-Band. G-Band views the error of each point along the line segment and models the quantity and direction of the error at each point to indicate an error ellipse of the point. Finally composition of these error ellipses construct the G-Band. Sample band geometry constructed for a line segment can be seen in Figure 6.4 to distinguish the geometric differences between G-Band and other models which are explained in the following sections.

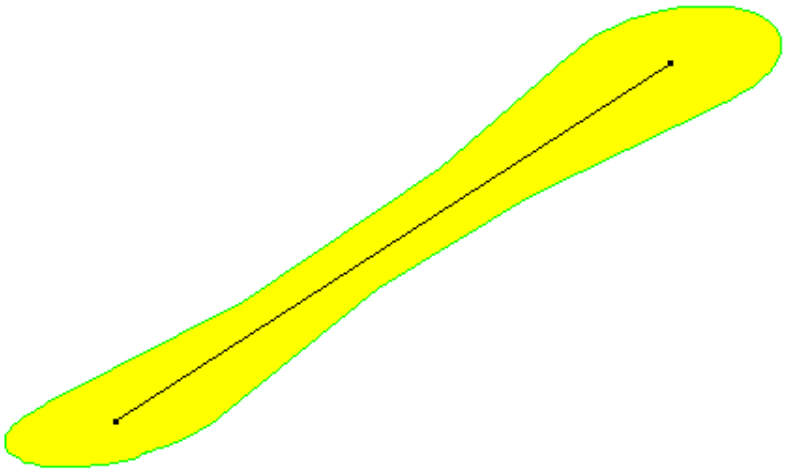


Figure 6.4: G-Band geometry constructed for a line segment

Developed software serves the implementation of G-Band model for the given sample dataset and visualizes the results by the help of GeoKIT Explorer.

6.1.2 Line Based Propagated Error Band (LPBM) Implementation

Detailed formulation and explanation of the model is given on Chapter 2. This model is mainly based on basic measurement error model with the law of error propagation. Over the knowledge of measurement error and error propagation, model formulate a simple, unified and effective treatment of error bands for a line segment, under the name of *covariance-based error model* which is similar to G-Band model (Leung et al., 2004).

Additional to G-Band model this model tries to model the error propagation through the nodes of the line segment. Propagation of the error from start node to the end node can be derived using a function which represents the positional relation between these nodes. In this thesis, propagation of error from start node to end

node of the network is modelled by a distance and a direction formula explained in Section 2.4.2, which helps to functionate the positional relation between end points of the line segment. Error which propagates through line segment, increases the end point's uncertainty. For this reason, differing from G-Band, in this model generated band expands at the end point of the line segment. A stand alone band generated for a line segment can be seen in Figure 6.5 to realize the band geometry changes between G-Band and LPBM.

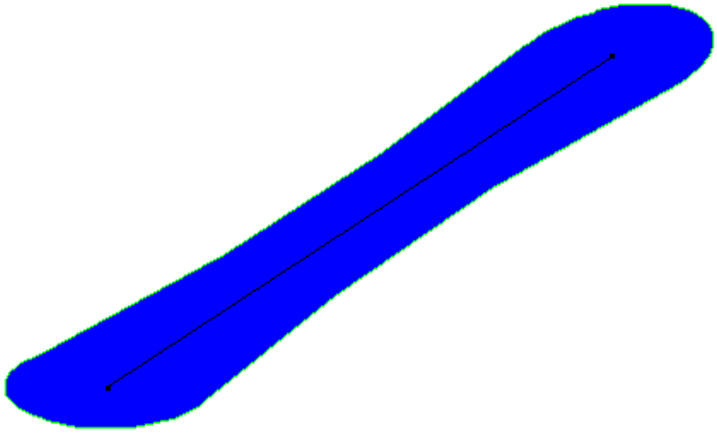


Figure 6.5: LPBM band geometry constructed for a line segment

In this alternative, each line segment that forms the network is handled separately and constructed band for each line segment is visualized by the help of GeoKIT Explorer. .

6.1.3 Network Based Propagated Error Band (NPBM) Implementation

In this alternative, although the line based calculation is the same as in Section 6.1.2, this time error propagation is brought to the ongoing line of the network. To realize this propagation for each line, end coordinates are recalculated by using the true length (l) and angle(θ) values which are derived from the true location of the line.

Following this, using the formulas expressed in objective part of the thesis are used to calculate new end coordinates sample set. The uncertainty of the start node for the following line is expressed using the sample set of recalculated end coordinates and process repeats until the end of the network is reached. Proposed method

considers the error which propagates from the start node of the previous line by applying this operation continuously for each line of the network. Finally this new method models the propagation of uncertainty for the whole network feature.

Figure 6.6 represents the band generated by NPBM. As NPBM model is not a segment based approach, Figure 6.6 is a visualization for an extracted segment from the network feature.

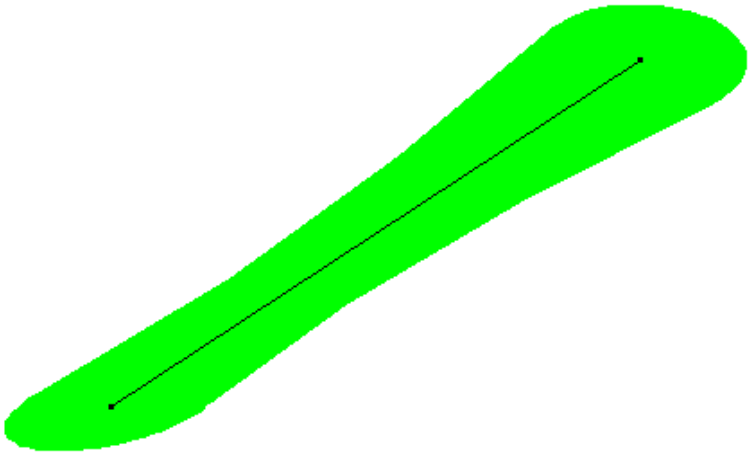
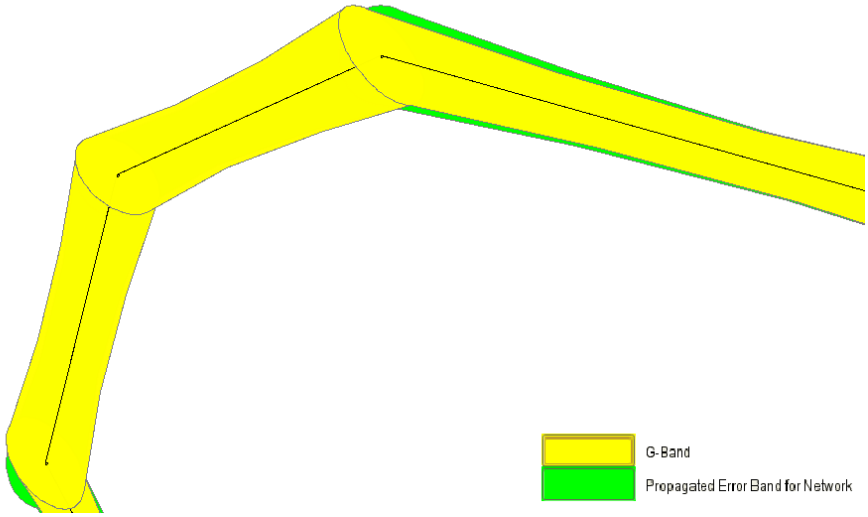
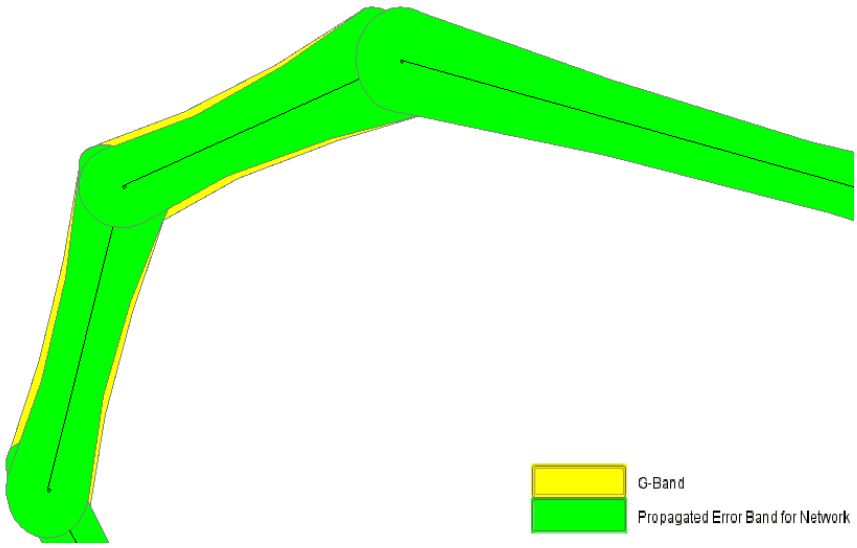


Figure 6.6: NPBM band geometry sample

NPBM band generally constructs bands with wider width than G-band for both start and end nodes of the line. But for some parts of the network at which the user selects closer points to the true value, band width of the NPBM band is narrower than G-Band. Both of these conditions can be viewed in Figures 6.7(a) and 6.7(b). Finally constructed bands for the network are visualized by the help of GeoKIT Explorer.



(a) G-Band over NPBM



(b) G-Band under NPBM

Figure 6.7: Difference between G-Band and NPBM in case of G-band is over NPBM(a) and G-Band is under NPBM (b)

6.1.4 Comparison of Implementations

In order to see the similarities and differences between stated models, developed software offers some alternatives for comperative analysis . User can select the alternative for the needed analysis from the wizard page as seen in Figure 6.8. Results are then displayed in GeoKIT Explorer.

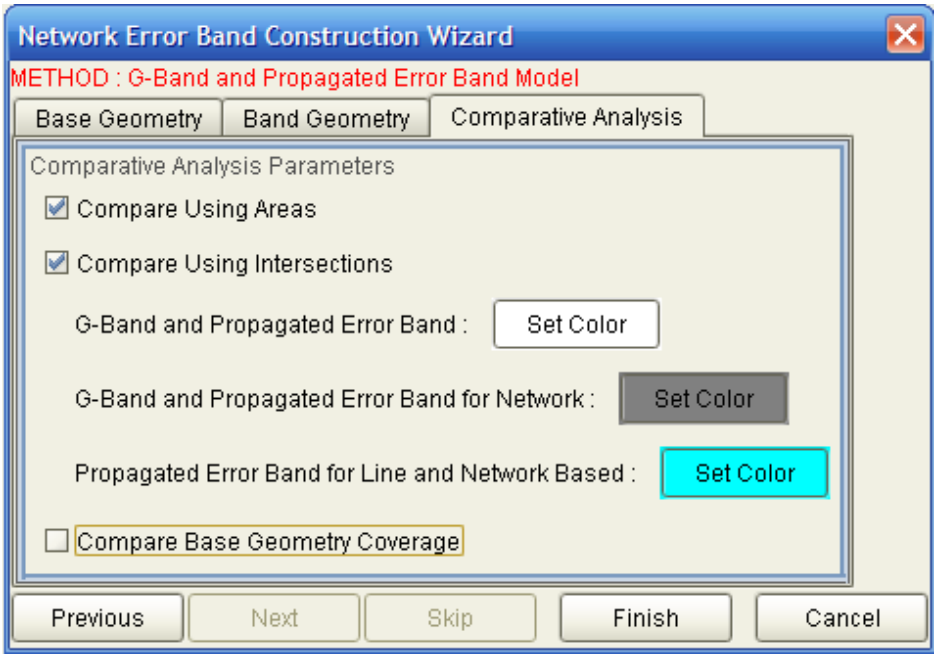


Figure 6.8: Comperative Analysis Parameter Selection Wizard Page of PBM Plug-in

6.1.4.1 Comparison of Band Geometry Area

Area values which are calculated for line segments separately for each implemented band model can be seen in Tables D.1 and D.2. Also totals are illustrated in Table D.3. Here the total difference between G-Band and NPBM can easily be distinguished but if the band areas are examined line by line, it can be realized that generated band areas of NPBM for some line segments are less than the band area of G-band. Main reason for this is the implementation of the algorithm (explained in Section 2.4.2) which is used to model the error propagation. While sample data is collected from the user, neither angle nor distance parameters which are used in error propagation algorithm in Section 2.4.2, is assumed to be fixed. For this reason, user can con-

sciously or unconsciously correct the error at any node of the network. To examine how error propagates through the network, user should not be allowed to correct the error during the digitizing process. For this reason, either angle or distance parameter has to be fixed to force the user to transmit the error of start node to ongoing nodes of the network during digitizing. But under normal conditions, in digitizing process this kind of limitations are not present and do not reflect the realism of digitizing. During digitizing process, user is not limited and allused to correct the starting error on the continuing nodes of the network. Although the uncertainty which propagates from start node and the generated band is expected to expand at the ongoing nodes of the network during uncertainty propagation modelling, dependent to user corrections explained above, for some nodes of the network generated NPBM bands are narrower than G-Band. This narrowing parts of the propagated error band can be noticed from sample data by using Sample Comparator Plug-in (see Section 4.3) to examine the convergence to true values.

Working principles of the algorithm that models the error propagation can be viewed in line based implementation of the propagated error band. For each line, error at the start node of the line segment is carried to the end. This propagation increases the error of the end point and cause the generated band to expand at the end of the line segment. This expansion can be noticed from the calculated area values illustrated in Tables D.1 and D.2. Also graphical difference between G-Band and line based Propagated Error can be seen in Figure 6.9.

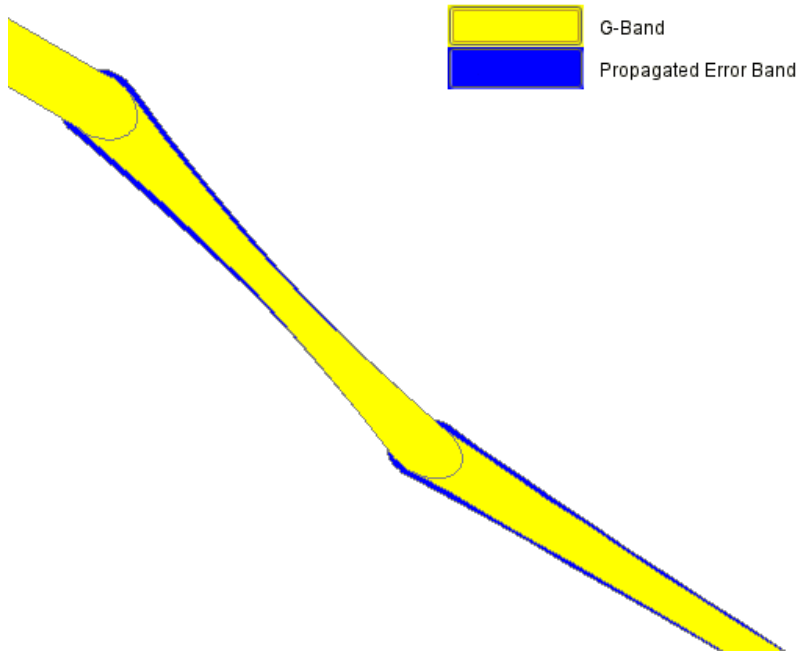


Figure 6.9: Difference between G-Band and Line Based Propagated Error Band

To conclude, the overall band area generated by NPBM approach is larger than both G-Band and LPBM models and if the sample coverage and band areas are compared in line based approach, it can be realized that NPBM generates better results than G-band approach for most of the lines of the network. For the condition explained in the paragraphs of this section above, if the user is forced to propagate the error through the nodes of the network, outcomes may seriously differ. This subject can be examined as future works.

6.1.4.2 Comparison of Intersection of Band Geometries

To reveal the difference between G-Band, Line Based and Network Based implementation of Propagated Error Band, generated band geometries are overlaid and intersection and difference operations are applied. Results are colored and displayed by GeoKIT Explorer. G-Band and Line Based Propagated Error Band, G-Band and Network Based Propagated Error Band, and finally Line Based and Network Based Propagated Error Band are overlaid respectively. Outcomes are not different from the results in Section 6.1.4.1.

- **G-Band and Line Based Propagated Error Band**

  - In Figure 6.10 G-Band geometry is represented by color *yellow* and color *blue* is used to represent LPBM. These two band geometries are overlaid and intersecting areas are colored with *white*. *Yellow* colored areas represent the G-Band residuals of difference operation, which is handled between G-Band and LPBM. Similiar to this, LPBM residuals are represented by color *blue*.

    In Figure 6.10, after the difference operation which is handled between LPBM and G-Band, it is found that LPBM residual area is larger at the end point of the line segment. This is expected because LPBM band has a larger band width at the end point of the line segment. Reason for this expansion is explained in Section 6.1.2.

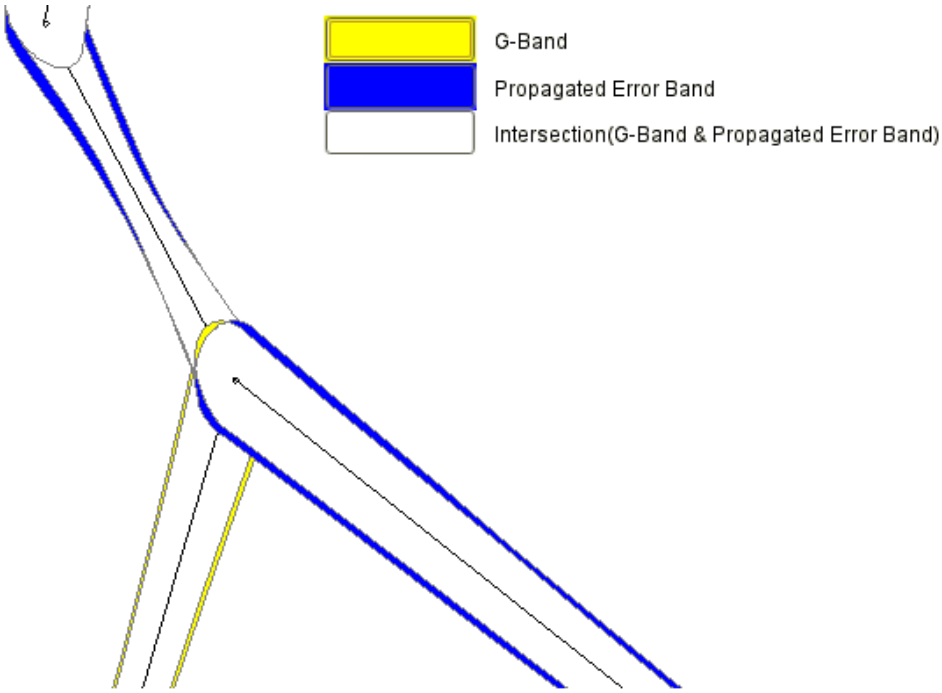


Figure 6.10: Intersection of G-Band and Line Based Propagated Error Band Geometries

- **G-Band and Network Based Propagated Error Band (NPBM)**

  – In Figure 6.11 G-Band geometry is represented by color *yellow* and color *green* is used to represent Network Based Propagated Error Band. These two band geometries are overlaid and intersecting areas are colored with *gray*. *Yellow* colored areas represent the G-Band residuals of difference operation which is handled between G-Band and NPBM. Similiar to this NPBM residuals are represented by color *green*.

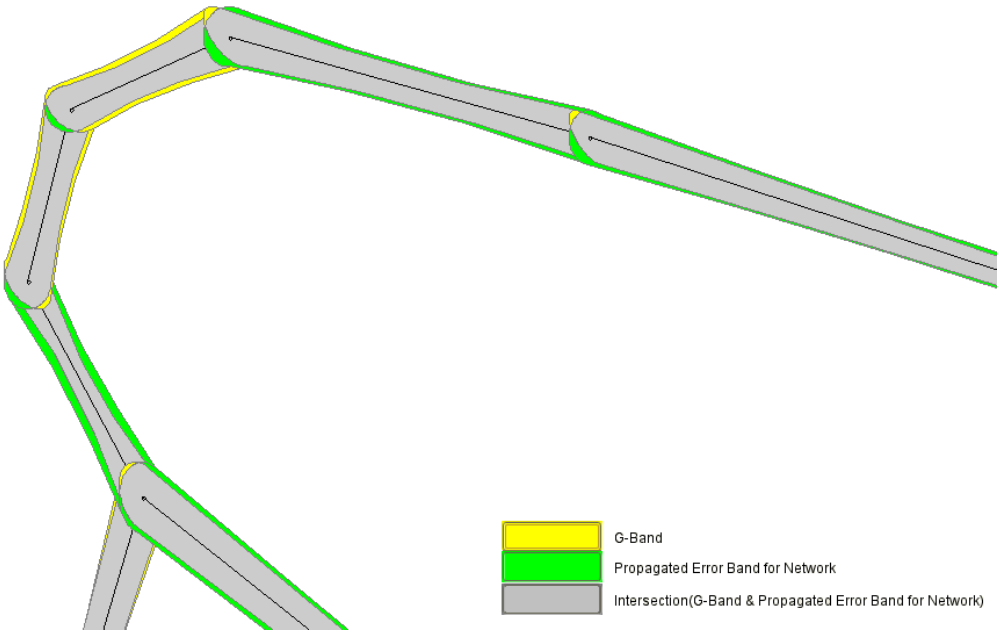


Figure 6.11: Intersection of G-Band and Network Based Propagated Error Band Geometries

In Figure 6.11, after the difference operation which is handled between NPBM and G-Band, it is found that NPBM residual areas are larger for both the start and end nodes of the line segment of the network. Also for some line segments there exists G-Band residuals after the difference operation of G-Band and NPBM. These residuals are sometimes so small to be noticed but sometimes clearly detected. Generated band geometries fully intersect for most of the network. Although the applied algorithm is different for both G-Band and NPBM, similarities are expected for some parts of the network. Introduced corrections of the user during digitizing

affect the generated band area of NPBM, so for some line segments of the network, G-Band area is larger than NPBM band area.

- **Line Based Propagated Error Band(LPBM) and Network Based Propagated Error Band(NPBM)**

  – In Figure 6.12 LPBM geometry is represented by color *blue* and color *green* is used to represent (NPBM). These two band geometries are overlaid and intersecting areas are colored with *white*. *Blue* colored areas represent the LPBM residuals of difference operation which is handled between LPBM and NPBM. Similiar to this NPBM residuals are represented by color *green*.

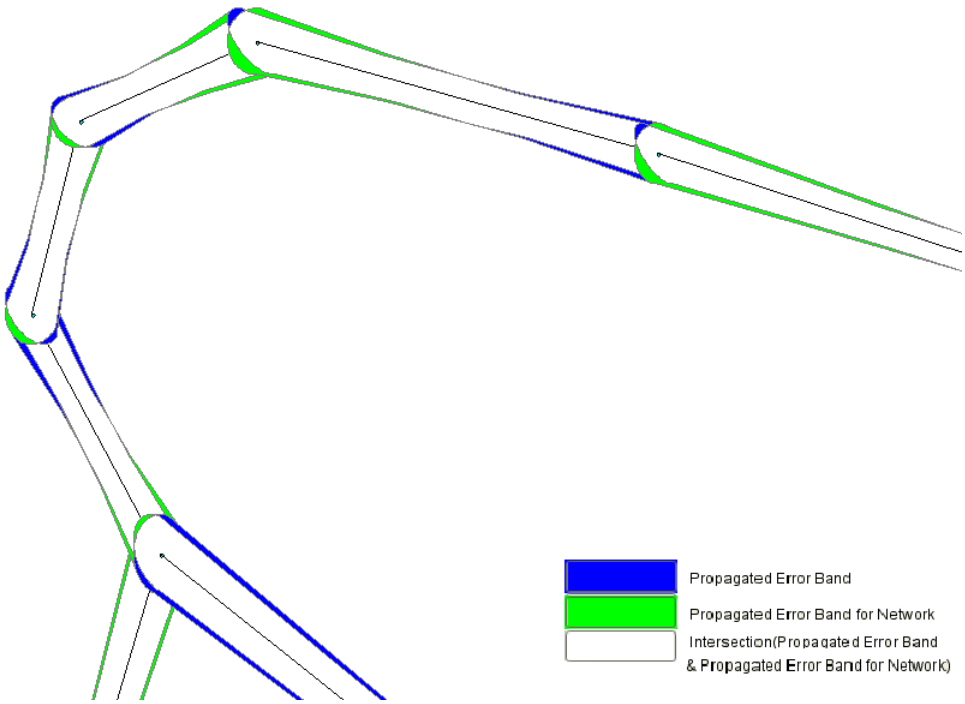


Figure 6.12: Intersection of Line and Network Based Propagated Error Band Geometries

LPBM and NPBM are different implementations of Measurement Error Model which is explained in Chapter 2. Here LPBM applies the error propagation algorithm explained in Section 2.4.2, for line segment based approach. For each line of the network this algorithm is run seperately and results are visualized. NPBM applies the algorithm in a generic way. It transfers the propagated error from line to line and try to model the error

propagation for the whole network. Differences between these two implementations are illustrated in Figure 6.12. For most of the network LPBM resudials (blue colored areas) are more visible on the segment ends but for line segments of shorter length and line segments which are elements of 3 or more line connections, NPBM residuals are more visible. Digitizing error on these line segments are greater (explained in Section 6.1.4.1) and adding propagated error makes the NPBM band to expand.

6.1.4.3 Comparison of Base and Digitized Geometry Coverage

Sample geometry coverage of each band model is illustrated separately for line segments in Tables D.4 and D.5. Also totals are illustrated in Table D.6. In Figure 6.13 numerical values given in Tables D.4 and D.5 are visualized.

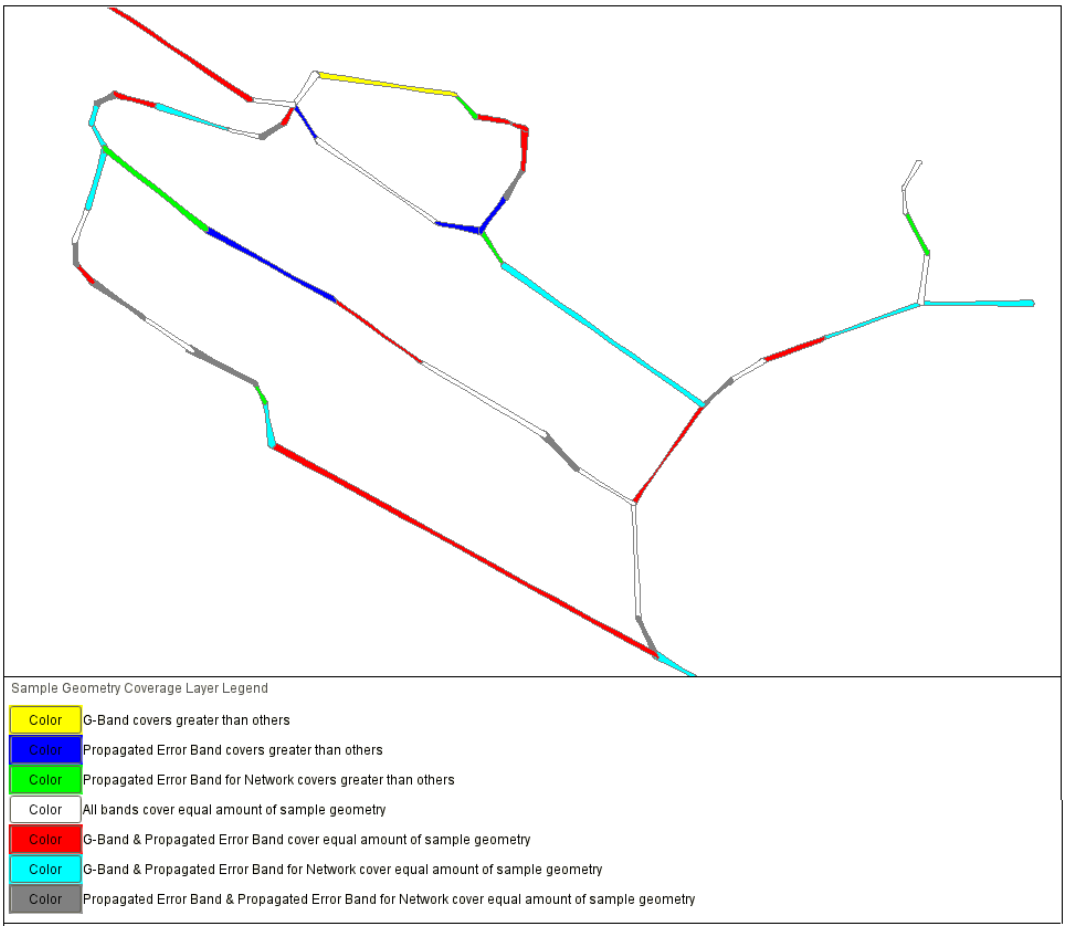


Figure 6.13: Visualization of Sample Geometry Coverage

Legend of Figure 6.13;

- G-Band covers greater than others : YELLOW
- Line Based Propagated Error Band(PBM) covers greater than others : BLUE
- Network Based PBM covers greater than others : GREEN
- All bands cover equal amount of sample geometry : WHITE
- G-Band and LPBM cover equal amount of sample geometry : RED
- G-Band and NPBM cover equal amount of sample geometry : CYAN
- LPBM and NPBM cover equal amount of sample geometry : GRAY

Finally, it is clear that NPBM approach covers most of the sample geometries. If the sample coverage and band areas are compared line by line, it can be noticed that NPBM generates better results than G-band approach for most of the segments of the network, as it has more coverage with smaller band area.

CHAPTER 7

CONCLUSIONS AND FUTURE WORKS

The basic aim of this thesis is to model uncertainty and its propagation in a network feature which is made up of line segments. G-Band model of Shi and Liu (2000) and Propagated Error Band Model of Leung et al. (2004) are applied band models which are used to analyze the uncertainty of the data. Model of Leung et al. (2004) is implemented in two different ways; line based implementation and network based implementation. For network based implementation, a new method is proposed for the implementation of the algorithm defined by Leung et al. (2004). In this method, rather than applying the algorithm in segment by segment, uncertainty which propagates from the previous line connected to the line of interest is considered. After applying these three different approaches to the selected network defined in Section 3.1.1, results are compared.

The comparisons are made by using band area, band geometry intersections and sample data coverage parameters for each band model applied to the network. Results are as follows;

- Comparison of G-Band and Line-Based Propagated Error Band (LPBM) approaches

  As mentioned before, error which propagates through line segment, cause an increase in end point's uncertainty. For this reason, different from G-Band, in LPBM, generated band expands at the end point of the line segment. This increase in the band geometry at the end point of the line affects both the band area and sample geometry coverage parameters. LPBM generates an area of 6137.05 unit squares for the whole network, however G-Band model has total band area of 5745.33 unit squares. LPBM generates nearly 6.4% greater band area than G-Band. For this reason, sample geometry covered by LPBM

is greater than G-Band. LPBM covers 543 of 780 sample lines, which is indicating a coverage of 69.9%. G-Band covers 535 of 780 sample lines which means 68.6% coverage for the sample data.

Finally, it is clear that for line segment based modeling of uncertainty, LPBM generates better results than G-Band model.

- Comparison of G-Band and Network-Based Propagated Error Band (NPBM) approaches

  As mentioned in the objective part of the thesis in Chapter 1, uncertainty propagation modeled with NPBM method may have varying shapes which are unpredictable. NPBM band generally constructs bands with wider width than G-band for both start and end nodes of the line. But for some parts of the network at which the user selects closer points to the assumed true value, band width of the NPBM band is narrower than G-Band. NPBM generates an area of 6692.14 unit squares for the whole network, however G-Band model has total band area of 5745.33 unit squares. NPBM generates nearly 14.1% greater band area than G-Band for the whole network. For this reason sample geometry covered by NPBM is greater than G-Band. NPBM covers 556 of 780 sample lines, which is indicating a coverage of 71.3%. G-Band covers 535 of 780 sample lines which means 68.6% coverage for the sample data.

  NPBM method's uncertainty modeling for some line segments of the network is better than G-Band approach because for some lines, although generated band area is less than G-Band, number of sample geometry coverage by NPBM is greater than G-Band model. Band areas generated for mentioned line segments with line IDs 3, 43 and sample geometry coverage can be viewed in Tables D.1, D.2, D.4 and D.5. For most of the line segments of the network, both generated band area and covered sample geometry is larger for NPBM than G-Band.

- Comparison of Line-Based and Network-Based Propagated Error Band approaches

  These two approaches are different implementations of the algorithm proposed by Leung et al. (2004). LPBM is a segment based approach which derives uncertainty propagation for each line segment of the network separately. However NPBM considers the uncertainty which is originated from the previous

line connected to the line of interest. When LPBM and NPBM is compared according to constructed band shapes, it can be seen that generated bands of LPBM has larger band width at the end node of the line segment. Although generated band geometry of NPBM has varying shapes for most of the line segments, generated band near to start point of the line have larger band width than LPBM. This increase in the band geometry affects both the band area and sample geometry coverage. LPBM generates an area of 6137.05 unit squares for the whole network, however NPBM model has total band area of 6692.14 unit squares. NPBM generates nearly 8.3% greater band area than LPBM. For this reason sample geometry covered by NPBM is greater than LPBM. LPBM covers 543 of 780 sample lines, which is indicating a coverage of 69.9%. NPBM covers 556 of 780 sample lines which means 71.3% coverage for the sample data.

LPBM and NPBM can both be used to model uncertainty propagation in a network feature. When line segment based approach is needed, LPBM can be preferred. When segment based uncertainty modeling techniques are compared. LPBM offers better results than G-Band. In case of uncertainty propagation modeling for the whole network feature, NPBM offers the best results for modeling the uncertainty which is related to human digitizing error.

Proposed method which models the uncertainty propagation in a network of line segments is important since modeling of uncertainty propagation in a network feature will form the basis of uncertainty prediction in network based analysis techniques in GIS such as route optimization, connectivity analysis, etc. Modeling uncertainty propagation for network analysis techniques should be the future work.

For instance, when shortest path is analyzed through a network of line segments, line length of the each segment requires a great importance. Derived results for route optimization is strictly related to the length of the line segments of interest and depending on this, the defined uncertainty region for that line segment. If a confidence region can be determined for the line segments of the whole network, then a certain confidence interval can be expressed for the length of the optimized route.

REFERENCES

Ayyub, B. and R.-J. Chao (1997). *"Uncertainty Modeling in Civil Engineering with Structural and Reliability Applications," Uncertainty Modeling and Analysis in civil engineering*, pp. pp. 3–32. New York: CRC Press.

Bastin, L., P. Fisher, and J. Wood (2002). Visualizing uncertainty in multi-spectral remotely sensed imagery. *Computers and Geosciences vol. 28*, pp. 337–350.

Bellevue_Community_College (2007). http://scidiv.bcc.ctc.edu/physics/measure&sigfigs/b-acc-prec-unc.html, last accessed: August 28.

Bilgi_GeoKIT (2007). http://geokit.bilgigis.com, last accessed: August 23.

Blenkinsop, S., P. Fisher, L. Bastin, and J. Wood (2000). Evaluating the perception of uncertainty in alternative visualization strategies. *Cartographica vol. 37*, pp. 1–13.

Bley, D. and R. Haller (2006). Checking the spatial accuracy of class boundaries with a varying range of accuracy requirements. In *7th International Symposium on Spatial Accuracy in Natural Resource and Environmental Sciences*, Lisboa - Portugal, pp. pp. 249–263.

Bonin, O. (2006). Sensitivity analysis and uncertainty analysis for vector geographical applications. In *7th International Symposium on Spatial Accuracy in Natural Resource and Environmental Sciences*, Saint-Mandé - CEDEX, pp. pp. 319–328.

Burrough, P. and R. McDonnel (1997). *Principles of Geographical Information Systems for Land Resource Assessment*. Oxford: Clarendon Press.

Caspary, W. and R. Scheuring (1992). Error-bands as measures of geometrical accuracy. *EGIS 92*, pp. 226–233.

Choi, H.-J. (2005, December). *A Robust Design Method for Model and Propagated Uncertainty*. Ph. D. thesis, Georgia Institute of Technology - School of Mechanical Engineering.

Chrisman, N. (1982). A theory of cartographic error and its measurement in digital data base. In *Proceedings of Auto Carto*, pp. pp. 159–168.

Dutton, G. (1992). Handling positional uncertainty in spatial databases. *Proceedings Spatial Data Handling Symposium 5 vol. 2*, pp. 460–469.

Fisher, P. (1991). Modelling soil map-unit inclusions by monte-carlo simulation. *International Journal of Geographical Information Systems vol. 5*, pp. 193–208.

Fisher, P. (1999). Models of uncertainty in spatial data. In P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind (Eds.), *Geographical Information Systems: Principles, Techniques, Management and Applications* (2 ed.), Volume vol. 1, Chapter 13, pp. pp. 191–203. New York: Wiley.

Foody, G. M. (1996). Approaches for production and evaluation of fuzzy land cover classifications from remotely sensed data. *International Journal of Remote Sensing vol.17*, pp. 1317–1340.

Genst, W. D. and F. Canters (2000). Handling uncertainty propagation through the buffer operation in a raster environment. In *In Proceedings Accuracy*, pp. pp. 145–152.

Goodchild, M. and G. Hunter (1997). A simple positional accuracy measure for linear features. *International Journal of Geographical Information Systems vol. 11*(3), pp. 299–306.

Goodchild, M. F., B. O. Parks, and L. T. Steyaert (1993). *Environmental Modeling with GIS*. Oxford: Oxford University Press.

Heuvelink, G., P. Burrough, and A. Stein (1989). Propagation of errors in spatial modelling with gis. *International Journal of Geographical Information Systems vol. 3*(4), pp. 303–322.

Heuvelink, G. B. M. (1993). *Error propagation in quantitative spatial modelling applications in Geographical Information Systems*. Ph. D. thesis, Utrecht University.

Heuvelink, G. B. M. (1998). *Error propagation in environmental modelling with GIS*. London: Taylor and Francis.

Heuvelink, G. B. M. and P. A. Burrough (1993). Error propagation in cartographic modelling using boolean and continuous classification. *International Journal of Geographical Information Systems vol. 7*(4), pp. 231–246.

King, J. P. (2000, December). Modeling boundaries of influence among positional uncertainty fields. Master's thesis, The Graduate School of The University of Maine.

Klir, G. J. and B. Yuan (1995). *Fuzzy Sets and Fuzzy Logic*. Upper Saddle River, NJ: Prentice Hall.

Kurtar, A. K. (2006, January). Uncertainty models for vector based functional curves and assessing the reliability of g-band. Master's thesis, Middle East Technical University - Geodetic and Geographic Information Technologies Department.

Leung, Y., J.-H. Ma, and M. F. Goodchild (2004). A general framework for error analysis in measurement-based gis part1:the basic measurement-error model and related concepts. *International Journal of Geographic Information Science vol. 14*(1), pp. 51–66.

Lilburne, L., D. Gatelli, and S. Tarantola (2006). Sensitivity analysis on spatial models: a new approach. In *7th International Symposium on Spatial Accuracy in Natural Resource and Environmental Sciences*, pp. pp. 329–338.

Lucieer, A. and M.-J. Kraak (2003). Interactive and visual fuzzy classification of remotely sensed imagery for exploration of uncertainty. *International Journal of Geographical Information Science vol. 18*(5), pp. 491–512.

Mas, J.-F. (2006). Reducing positional error in spatio-temporal analyses. In *7th International Symposium on Spatial Accuracy in Natural Resource and Environmental Sciences*, Lisboa - Portugal, pp. pp. 281–285.

Montana_University (2007). http://www.forestry.umt.edu/academics/courses/for503/gis_errors.htm, last accessed: August 13.

Navratil, G., M. Franz, and E. Pontikakis (2004). Measurement-based gis revisited. In *7th AGILE Conference on Geographic Information Science: 29 April-1 May*, Heraklion, Greece, pp. pp. 771–775.

Neuilly, M. and CETAMA (1999). *Modelling and estimation of measurement errors*. Hampshire(UK) and New York: Intercept Ltd.

Niel, T. G. V. and T. R. McVicar (2002). Experimental evaluation of positional accuracy estimates from a linear network using point- and line-based testing methods. *International Journal of Geographic Information Science vol. 16*(5), pp. 455–473.

Perkal, J. (1956). On epsilon length. *Bulletin de I'Academie Polonaise des Sciences vol. 4*, pp. 399–403.

Perkal, J. (1966). On the length of empirical curves. *Michigan Inter-University Community of Mathematical Geographers*.

Rabinovich, S. G. (2000). *Measurement Errors and Uncertainties Theory and Practice*. New York: Springer-Verlag.

Rodrigue, J.-P. (2007). http://people.hofstra.edu/geotrans/eng/ch2en/meth2en/ch2m3en.html, last accessed: September 1.

Shi, W. (1994). Modelling positional and thematic uncertainty in integration of gis and remote sensing. *ITC Publication* (22).

Shi, W., M. Goodchild, and P. Fisher (1999). Measurement-based gis. In *Proceedings of the International Symposium on Spatial Data Quality '99*, Hong Kong Polytechnic University, Hong Kong, pp. pp. 1–9.

Shi, W. and W. Liu (2000). A stochastic process-based model for the positional error of line segments in gis. *International Journal of Geographic Information Science vol. 14*(1), pp. 51–66.

Veregin, H. (1992). Development and test of an error model for categorical data. *International Journal of Geographical Information Systems vol. 6*, pp. 87–104.

Veregin, H. (1996). Error propagation through the buffer operation for probability surfaces. *Photogrammetric Engineering and Remote Sensing* (62), pp. 419–428.

Wang, F. (1990). Improving remote sensing image analysis through fuzzy information representation. *Photogrammetric Engineering and Remote Sensing vol. 56*, pp. 1163–1169.

Wel, F. V. D., L. V. D. Gaag, and B. Gorte (1997). Visual exploration of uncertainty in remote sensing classification. *Computers and Geosciences vol. 24*, pp. 335–343.

Wolf, P. and C. Ghilani (1997). *Adjustment Computations: Statistics and least squares in surveying and GIS*. New York: John Wiley.

WolframMathWorld_Line (2007). http://mathworld.wolfram.com/line.html, last accessed: August 28.

Yu, Z. C. and L. C. Lu (1983). *Fundamentals of Surveying Adjustment*. China: Beijing: China Surveying and Mapping.

Zadeh, L. (1965). Fuzzy sets. *Information and Control vol.8*, pp. 338–353.

Zhang, B., L. Zhu, and G. Zhu (1998). The uncertainty propagation model of vector data on buffer operations in gis. *ACTA Geodaetica et Cartographic Sinica* (27), pp. 259–266.

Zhang, J. and G. M. Foody (2001). Fully-fuzzy supervised classification of sub-urban land cover from remotely sensed imagery: statistical and artificial neural network approaches. *International Journal of Remote Sensing vol.22*, pp. 615–628.

Zhang, J. and M. F. Goodchild (2002). *Uncertainty in Geographical Information*. London: Taylor and Francis.

APPENDIX A

FILE FORMAT EXAMPLES

A.1 Sample File Format

Group ID,Feature ID, Feature Type, Start Node ID, Start X, Start Y, Reference Node ID(s) of Start, End Node ID, End X, End Y, Reference Node ID(s) of End

1,1,LINE,1,664,215,(),2,596,213,(42)

1,2,LINE,2,596,213,(42),3,539,235,()

1,3,LINE,3,539,235,(),4,503,246,()

1,4,LINE,4,503,246,(),5,487,258,()

1,5,LINE,5,487,258,(),6,470,274,(34)

1,6,LINE,6,470,274,(34),7,429,333,(35)

1,7,LINE,7,429,333,(35),8,431,399,()

1,8,LINE,8,431,399,(),9,444,423,(11)

1,9,LINE,9,444,423,(11),10,463,433,()

2,10,LINE,11,444,423,(9),12,214,295,()

3,11,LINE,12,214,295,(),13,212,272,()

3,12,LINE,13,212,272,(),14,205,261,()

3,13,LINE,14,205,261,(),15,166,242,()

3,14,LINE,15,166,242,(),16,140,223,()

3,15,LINE,16,140,223,(),17,110,200,()

3,16,LINE,17,110,200,(),18,101,191,()

3,17,LINE,18,101,191,(),19,99,175,()

3,18,LINE,19,99,175,(),20,108,156,()

3,19,LINE,20,108,156,(),21,118,121,(41)

.....

A.2 Simple Geometry File (SGF) Format

confidence=95.0

Group_Id: Feature_ID: Feature_Type(LINE):Point_Number(2 for LINE features):

{Point1_ID, Point1_X, Point1_Y, Ref_Node_Ids, Point1_MeanX, Point1_MeanY,

Point1_VarX, Point1_VarY, Point2_ID, Point2_X, Point2_Y, (Ref_Node_Ids),

Point2_MeanX, Point2_MeanY, Point2_VarX, Point2_VarY,

{GBAND-

[$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1$];

PROPAGATED_BAND_LINE-

[$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1$];

PROPAGATED_BAND_NETWORK-

[$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1$]}}

Here is an example for SGF:

1:1:LINE:2:

{1, 663.0, 214.0, (), 663.0666666666667,213.86666666666667, 0.961150104723255,

1.0600988273786194, 2, 598.0, 214.0, (),597.7333333333333, 213.73333333333332,

0.9611501047232548, 0.5936168397046637,{

GBAND-

[0.01904761904761901 | 0.22380952380952382 | 0.019047619047619067

| 0.10476190476190472 | 0.280952380952381 | −0.038095238095238064];

PROPAGATED_BAND_LINE-

[0.01904761904761901 | 0.22380952380952382 | 0.019047619047619067

| 0.10476190476190472 | 4.017772351148417 | −0.038095238095238064];

PROPAGATED_BAND_NETWORK-

[0.01904761904761901 | 0.22380952380952382 | 0.019047619047619067

| 0.10476190476190472 | 4.017772351148417 | −0.038095238095238064]}}

. . . . . .

A.3 Complex Geometry File (CGF) Format

Organization of the file can be seen below. Here attributes of each geometric feature are represented.

confidence=95.0

LINE Feature

Feature_ID: Feature_Type(LINE):PointNumber(2 for LINE feature):

{Point1_ID, Point1_X, Point1_Y, Point1_MeanX, Point1_MeanY,

Point1_VarX, Point1_VarY, Point2_ID, Point2_X, Point2_Y,

Point2_MeanX, Point2_MeanY, Point2_VarX, Point2_VarY,

{[Variance-Covariance Matrix_Point12:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1]\}\}$

ARC Feature

Feature_ID: Feature_Type(ARC): PointNumber(3 for ARC feature):

{Point1_ID, Point1_X, Point1_Y, Point1_MeanX, Point1_MeanY,

Point1_VarX, Point1_VarY, Point2_ID, Point2_X, Point2_Y,

Point2_MeanX, Point2_MeanY, Point2_VarX, Point2_VarY,

Point3_ID, Point3_X, Point3_Y, Point3_MeanX, Point3_MeanY,

Point3_VarX, Point3_VarY,

{[Variance-Covariance Matrix_Point12:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1]$;

[Variance-Covariance Matrix_Point23:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1]\}\}$

SPLINE Feature

Feature_ID: Feature_Type(SPLINE): PointNumber(4 for SPLINE feature):

{Point1_ID, Point1_X, Point1_Y, Point1_MeanX, Point1_MeanY,

Point1_VarX, Point1_VarY, Point2_ID, Point2_X, Point2_Y,

Point2_MeanX, Point2_MeanY, Point2_VarX, Point2_VarY,

Point3_ID, Point3_X, Point3_Y, Point3_MeanX, Point3_MeanY,

Point3_VarX, Point3_VarY, Point4_ID, Point4_X, Point4_Y,

Point4_MeanX, Point4_MeanY, Point4_VarX, Point4_VarY,

{[Variance-Covariance Matrix_Point12:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1]$;

[Variance-Covariance Matrix_Point23:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1$];

[Variance-Covariance Matrix_Point34:

$covX0X1 \mid covX0Y0 \mid covX0Y1 \mid covX1Y0 \mid covX1Y1 \mid covY0Y1$]}}

Here is a sample CGF file named test.cgf;

confidence=95.0

1:LINE:2:

{1,108,102,107.56,102.05,0.916,0.928,2,99,191,98.77,190.84,0.864,

0.851,{

[$0.2 \mid 0.2 \mid 0.2 \mid 0.2 \mid 0.2 \mid 0.2$]}}

2:SPLINE:4:

{3,596,129,595.77,128.84,0.864,0.851,4,588,158,587.87,158.24,0.854,

0.861,5,601,182,601.52,181.94,0.884,0.867,6,598,213,597.87,213.34,

0.844,0.873,{

[$0.2 \mid 0.2 \mid 0.2 \mid 0.2 \mid 0.3 \mid 0.3$];

[$0.3 \mid 0.4 \mid 0.1 \mid 0.1 \mid 0.2 \mid 0.3$];

[$0.1 \mid 0.2 \mid 0.3 \mid 0.1 \mid 0.2 \mid 0.2$]}}

3:ARC:3:

{7,119,122,118.87,122.34,0.854,0.867,8,110,177,110.87,107.34,0.874,

0.851,9,112,96,112.52,95.94,0.824,0.867,{

[$0.1 \mid 0.2 \mid 0.1 \mid 0.2 \mid 0.1 \mid 0.2$];

[$0.2 \mid 0.1 \mid 0.1 \mid 0.2 \mid 0.2 \mid 0.1$]}}

. . . . . .

APPENDIX B

PROPAGATED BAND MODEL DRAWER SYSTEM CODE SAMPLES

B.1 IPBMNetworkFeature Interface

IPBMNetworkFeature interface which is implemented by supported geometries represented by *PBMSampleLine*, *PBMSampleArc*, *PBMSampleSpline* and for network based feature objects *PBMNetworkLine*, *PBMNetworkArc* and *PBMNetworkSpline*.

```
 * Created on 03-Mar-2007
package metu.ggit.thesis.pbm.network.model;

 * @author OYA YARKINOGLU GUCUK

public interface IPBMNetworkFeature {
    //Possible feature type identifiers;
    public static final String LINE = "LINE";
    public static final String ARC = "ARC";
    public static final String SPLINE = "SPLINE";


    public int getGroupID();

    public int getFeatureID();

    public String getFeatureTypeStr();
}
```

B.2 ICGFFileObject Interface

ICGFFileObject interface which is implemented by supported geometries represented by *CGFLineFeature*, *CGFArcFeature*, *CGFSplineFeature* and *SGFLineFeature*.

```
package metu.ggit.thesis.pbm.reader.cgf.model;

import com.bilgi.geokit.kernel.feature.geom.GKLinearRing;
import com.metu.ggit2006.kursat.thesis.statistics.CovarianceHolder;

public interface ICGFFileObject {

    public static int LINE_TYPE = 0;
    public static int ARC_TYPE = 1;
    public static int SPLINE_TYPE = 2;

    public int getFeatureType();
    public void setFeatureType(int featureType);
    public double[] getMeansX();
    public double[] getMeansY();
    public double[] getVarianceX();
    public double[] getVarianceY();
    public double[] getStDeviationX();
    public double[] getStDeviationY();
    public void setMeansX(double[] meansX);
    public void setMeansY(double[] meansY);
    public void setVarianceX(double[] variancesX);
    public void setVarianceY(double[] variancesY);
    public void setStDeviationX(double[] stDevsX);
    public void setStDeviationY(double[] stDevsY);
    public double getConfidence();
    public void setConfidence(double confidence);
    public CovarianceHolder[] getCovarianceHolderArray();
    public void setCovarianceHolderArray(CovarianceHolder[] covarianceHolderArr);
    public GKLinearRing getConstructedGBand();
}
```

B.3 SGFFileReader Class

SGFFileReader parses the content of the files having an extension of *.SGF. *SGFFileReader* constructor and responsible function for parsing the content can be seen in the below figures respectively.

```
package metu.ggit.thesis.pbm.reader.sgf;

import java.io.BufferedReader;

public class SGFFileReader {
    BufferedReader reader;
    private Vector<ICGFFileObject> sgfFileContent;

    public SGFFileReader(String selectedDirectory, String selectedMapName) {
        String filePath = selectedDirectory + "/" + selectedMapName;
        //create the file and open a stream to read it
        try {
            File sgfFile = new File(filePath);
            reader = new BufferedReader(new FileReader(sgfFile));
            readFileContent();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    public Vector<ICGFFileObject> getFileContent() {
        return sgfFileContent;
    }

     ............
```

Code block which parses the file content. Function **readFileContent()** :

```
private void readFileContent() {
    try {
        sgfFileContent = new Vector<ICGFFileObject>();
        //Read confidence value that will be used during band construction
        String confidenceStr = reader.readLine();
        double confidence = Double.parseDouble(
                confidenceStr.substring(11, confidenceStr.length()));
        //read the data
        String lineRead = reader.readLine();
        while(lineRead != null){
            //tokenizer the read line
            StringTokenizer tokenizer = new StringTokenizer(lineRead,":");
            while(tokenizer.hasMoreTokens()){
                int groupId = Integer.parseInt(tokenizer.nextToken());
                int featureId = Integer.parseInt(tokenizer.nextToken());
                String featureType = tokenizer.nextToken();
                int numCoords = Integer.parseInt(tokenizer.nextToken());
                String coordinate_Stats_Str = tokenizer.nextToken();

                SGFLineFeature sgfLineFeature = new SGFLineFeature();
                sgfLineFeature.setFeatureType(ICGFFileObject.LINE_TYPE);
                sgfLineFeature.setConfidence(confidence);

                fillSGFObjectContent(groupId, featureId, numCoords,
                        coordinate_Stats_Str, sgfLineFeature);
                sgfFileContent.addElement(sgfLineFeature);
                lineRead = reader.readLine();
            }
        }
    } catch (Exception e) {
        System.err.println("ERROR reading SGF file content...");
        e.printStackTrace();
    }
}
```

B.4 CGFFileReader Class

CGFFileReader parses the content of the files having an extension of *.CGF. *CGFFileReader* constructor and responsible function for parsing the content can be seen in the below figures respectively.

```
public class CGFFileReader {

    BufferedReader reader;
    private Vector<ICGFFileObject> cgfFileContent;

    public CGFFileReader(String selectedDirectory, String selectedMapName) {
        String filePath = selectedDirectory + "/" + selectedMapName;
        //create the file and open a stream to read it
        try {
            File cgfFile = new File(filePath);
            reader = new BufferedReader(new FileReader(cgfFile));
            readFileContent();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    public Vector<ICGFFileObject> getFileContent() {
        return cgfFileContent;
    }

    .............
```

```java
private void readFileContent() {
    try {
        cgfFileContent = new Vector<ICGFFileObject>();
        //Read confidence value that will be used during band construction
        String confidenceStr = reader.readLine();
        double confidence = Double.parseDouble(
                confidenceStr.substring(11, confidenceStr.length()));
        //read the data
        String lineRead = reader.readLine();
        while(lineRead != null){
            //tokenizer the read line
            StringTokenizer tokenizer = new StringTokenizer(lineRead,":");
            while(tokenizer.hasMoreTokens()){
                int featureId = Integer.parseInt(tokenizer.nextToken());
                String featureType = tokenizer.nextToken();
                int numCoords = Integer.parseInt(tokenizer.nextToken());
                String coordinate_Stats_Str = tokenizer.nextToken();
                //HANDLE THE FEATURES
                if(featureType.equals(IPBMNetworkFeature.LINE)){
                    CGFLineFeature cgfLineFeature = new CGFLineFeature();
                    cgfLineFeature.setFeatureType(ICGFFileObject.LINE_TYPE);
                    cgfLineFeature.setConfidence(confidence);
                    fillCGFObjectContent(featureId, numCoords,
                            coordinate_Stats_Str, cgfLineFeature);
                    cgfFileContent.addElement(cgfLineFeature);
                }else
                    if(featureType.equals(IPBMNetworkFeature.ARC)){
                        CGFArcFeature cgfArcFeature = new CGFArcFeature();
                        cgfArcFeature.setFeatureType(ICGFFileObject.LINE_TYPE);
                        cgfArcFeature.setConfidence(confidence);
                        fillCGFObjectContent(featureId, numCoords,
                                coordinate_Stats_Str, cgfArcFeature);
                        cgfFileContent.addElement(cgfArcFeature);
                    }else
                        if(featureType.equals(IPBMNetworkFeature.SPLINE)){
                            CGFSplineFeature cgfSplineFeature = new CGFSplineFeature();

                            cgfSplineFeature.setFeatureType(ICGFFileObject.LINE_TYPE);
                            cgfSplineFeature.setConfidence(confidence);
                            fillCGFObjectContent(featureId, numCoords,
                                    coordinate_Stats_Str, cgfSplineFeature);
                            cgfFileContent.addElement(cgfSplineFeature);
                        }else{
                            new Exception("UNKNOWN FEATURE TYPE!!!");
                        }
            }//end of line tokenizer WHILE loop
            lineRead = reader.readLine();
        }
    } catch (Exception e) {
        System.err.println("ERROR reading CGF file content...");
        e.printStackTrace();
    }
```

B.5 PBMBandStatCalculator Class

PBMBandStatCalculator calculates the necessary statistic used during band generation.

G-BAND

G-band statistics and band generation are activated by function

public PBMNetworkLine ***calculateStatistics****(GKCoordinateArray arrayStartNode, GKCoordinateArray arrayEndNode, int index, PBMSampleLine originalLine, boolean baseGiven)*

Java code of the method can be seen in Figure B.1;

LINE-BASED PROPAGATED ERROR BAND

Line-based propagated error band statistics and band generation are activated by function

public void ***calculatePropagatedStats4Line****(PBMNetworkLine nLine, GKCoordinateArray arrayAngleLength, GKCoordinateArray arrayNewEnd)*

Java code of the method can be seen in Figure B.2;

NETWORK-BASED PROPAGATED ERROR BAND

Network-based propagated error band statistics and band generation are activated by function

private void ***calculatePropagatedStats4Network****(PBMNetworkLine nLine, GKCoordinateArray arrayAngleLength,GKCoordinateArray arrayAngle, GKCoordinateArray arrayLength)*

Java code of the method can be seen in Figure B.3;

```java
public PBMNetworkLine calculateStatistics(GKCoordinateArray arrayStartNode,
        GKCoordinateArray arrayEndNode, int index, PBMSampleLine originalLine,
        boolean baseGiven) {
    //Calculate point statistics
    Sampler sStart = new Sampler(arrayStartNode);
    Sampler sEnd = new Sampler(arrayEndNode);
    double startStdevX = sStart.getStandardDeviationX();
    double startStdevY = sStart.getStandardDeviationY();
    double startCVX = startStdevX / sStart.getMeanx();
    double startCVY = startStdevY / sStart.getMeany();

    double endStdevX = sEnd.getStandardDeviationX();
    double endStdevY = sEnd.getStandardDeviationY();
    double endCVX = endStdevX / sEnd.getMeanx();
    double endCVY = endStdevY / sEnd.getMeany();

    PBMSampleNode startNode = originalLine.getStart();
    PBMSampleNode endNode = originalLine.getEnd();

    //calculate covariance of start and end points
    CovarianceFinder covarianceFinder = new CovarianceFinder(sStart, sEnd);
    UncertainPoint uStart;
    UncertainPoint uEnd;
    if(baseGiven){
        uStart = new UncertainPoint(originalLine.getStart().x, originalLine.getStart().y,
                sStart.getStandardDeviationX(), sStart.getStandardDeviationY(),
                covarianceFinder.getCovarianceHolder().getCovariancex0y0());
        uEnd = new UncertainPoint(originalLine.getEnd().x, originalLine.getEnd().y,
                sEnd.getStandardDeviationX(), sEnd.getStandardDeviationY(),
                covarianceFinder.getCovarianceHolder().getCovariancex1y1());
    }else{
        uStart = new UncertainPoint(sStart.getMeanx(), sStart.getMeany(),
                sStart.getStandardDeviationX(), sStart.getStandardDeviationY(),
                covarianceFinder.getCovarianceHolder().getCovariancex0y0());
        uEnd = new UncertainPoint(sEnd.getMeanx(), sEnd.getMeany(),
                sEnd.getStandardDeviationX(), sEnd.getStandardDeviationY(),
                covarianceFinder.getCovarianceHolder().getCovariancex1y1());
    }
    //Construct network line
    double z = ConfidenceFinder.findZ(confidence/100.0);
    PBMNetworkLine nLine = new PBMNetworkLine(uStart, uEnd, z);
    nLine.confidence = confidence;
    nLine.originalLine = originalLine;
    nLine.groupID = originalLine.getGroupID();
    nLine.lineID = originalLine.getFeatureID();
    nLine.startNodeId = originalLine.getStart().getNodeId();
    nLine.endNodeId = originalLine.getEnd().getNodeId();
    nLine.startRefNodes = originalLine.getStart().getRefNodes();
    nLine.endRefNodes = originalLine.getStart().getRefNodes();
    nLine.startSampler = sStart;
    nLine.endSampler = sEnd;
    nLine.covarianceFinder = covarianceFinder;
    //generate gBand
    try{
        LineGBand gBand = new LineGBand(nLine,nLine.covarianceFinder.getCovarianceHolder());
        nLine.gBand = gBand;
        return nLine;
    }catch (Exception e){
        return null;
    }
}
```

Figure B.1: G-Band statistics calculator and band generator method

```java
public void calculatePropagatedStats4Line(PBMNetworkLine nLine,
        GKCoordinateArray arrayAngleLength, GKCoordinateArray arrayNewEnd){
    GKCoordinateArray arrayAngle = new GKCoordinateArray();
    GKCoordinateArray arrayLength = new GKCoordinateArray();
    //divide the content of arrayaAngleLength. This is necessary for constructing
    //seperate samplers for angle and length to calculate covariance between
    for(int j = 0; j < arrayAngleLength.numCoordinates; j++){
        GKCoordinate alCoord = (GKCoordinate)arrayAngleLength.getCoordinate(j);
        arrayAngle.add(alCoord.x, 0.0);
        arrayLength.add(0.0, alCoord.y);}

    GKCoordinateArray arrayStartNode = nLine.startSampler.getCoordinates();
    Sampler samplerStart = new Sampler(arrayStartNode);
    Sampler samplerAngleLength = new Sampler(arrayAngleLength);
    Sampler samplerNewEndCoord = new Sampler(arrayNewEnd);
    nLine.samplerNewEndCoord = samplerNewEndCoord;
    Sampler samplerAngle = new Sampler(arrayAngle);
    Sampler samplerLength = new Sampler(arrayLength);
    CovarianceFinder finderAngLen = new CovarianceFinder(samplerAngle, samplerLength);
    CovarianceHolder holderAngLen = finderAngLen.getCovarianceHolder();

    //calculate propagated error statistics for line
    CovarianceFinder finder = new CovarianceFinder(samplerStart, samplerNewEndCoord);
    CovarianceHolder holderPropagated = finder.getCovarianceHolder();

    GKMatrix jacobian = new GKMatrix(2,2);
    jacobian.setValueAt(Math.cos(samplerAngleLength.getMeany()),0,0);
    jacobian.setValueAt((-samplerAngleLength.getMeanx() *
            Math.sin(samplerAngleLength.getMeany())),0,1);
    jacobian.setValueAt(Math.sin(samplerAngleLength.getMeany()),1,0);
    jacobian.setValueAt((samplerAngleLength.getMeanx() *
            Math.cos(samplerAngleLength.getMeany())),1,1);

    GKMatrix Em1 = new GKMatrix(2,2);
    Em1.setValueAt(samplerAngle.getVariancex(),0,0);
    Em1.setValueAt(holderAngLen.getCovariancex0y0(),0,1);
    Em1.setValueAt(holderAngLen.getCovariancex0y0(),1,0);
    Em1.setValueAt(samplerLength.getVariancex(),1,1);
    GKMatrix matrix1 = jacobian.multiply(Em1);

    GKMatrix propMatrix1 = matrix1.multiply(jacobian.transpose());
    CovarianceHolder holderPropagatedOriginal1 = new CovarianceHolder(
            holderPropagated.getCovariancex0x1(),holderPropagated.getCovariancex0y0(),
            holderPropagated.getCovariancex0y1(), holderPropagated.getCovariancex1y0(),
            propMatrix1.getValueAt(0,1),holderPropagated.getCovarianceyOy1());
    nLine.gBandPropagated = new LineGBand(nLine, holderPropagatedOriginal1);
    nLine.holderPropagated = holderPropagatedOriginal1;
}
```

Figure B.2: Line-based propagated error band statistics calculator and band generator method

```
private void calculatePropagatedStats4Network(PBMNetworkLine nLine,
        GKCoordinateArray arrayAngleLength,GKCoordinateArray arrayAngle,
        GKCoordinateArray arrayLength){
    Sampler samplerStart = nLine.samplerNewStartCoord4Network;
    Sampler samplerNewEndCoord = nLine.samplerNewEndCoord4Network;
    Sampler samplerAngleLength = new Sampler(arrayAngleLength);
    Sampler samplerAngle = new Sampler(arrayAngle);
    Sampler samplerLength = new Sampler(arrayLength);
    CovarianceFinder finderAngLen = new CovarianceFinder(samplerAngle, samplerLength);
    CovarianceHolder holderAngLen = finderAngLen.getCovarianceHolder();
    CovarianceFinder finder = new CovarianceFinder(samplerStart, samplerNewEndCoord);
    CovarianceHolder holderPropagated = finder.getCovarianceHolder();

    GKMatrix jacobian = new GKMatrix(2,2);
    jacobian.setValueAt(Math.cos(samplerAngleLength.getMeany()),0,0);
    jacobian.setValueAt((-samplerAngleLength.getMeanx() *
            Math.sin(samplerAngleLength.getMeany())),0,1);
    jacobian.setValueAt(Math.sin(samplerAngleLength.getMeany()),1,0);
    jacobian.setValueAt((samplerAngleLength.getMeanx() *
            Math.cos(samplerAngleLength.getMeany())),1,1);
    GKMatrix Em1 = new GKMatrix(2,2);
    Em1.setValueAt(samplerAngle.getVariancex(),0,0);
    Em1.setValueAt(holderAngLen.getCovariancex0y0(),0,1);
    Em1.setValueAt(holderAngLen.getCovariancex0y0(),1,0);
    Em1.setValueAt(samplerLength.getVariancex(),1,1);

    GKMatrix matrix1 = jacobian.multiply(Em1);
    GKMatrix propMatrix1 = matrix1.multiply(jacobian.transpose());


    CovarianceHolder holderPropagatedOriginal1 = new CovarianceHolder(
            holderPropagated.getCovariancex0x1(),holderPropagated.getCovariancex0y0(),
            holderPropagated.getCovariancex0y1(), holderPropagated.getCovariancex1y0(),
    propMatrix1.getValueAt(0,1),holderPropagated.getCovariancey0y1());
    nLine.gBandPropagated4Network = new LineGBand(nLine, holderPropagatedOriginal1);
    nLine.holderPropagated4Network = holderPropagatedOriginal1;
}
```

Figure B.3: Network-based propagated error band statistics calculator and band generator method

APPENDIX C

INTERACTIVE DRAWER CODE SAMPLES

C.1 Approve Action

```
if(e.getSource().equals(approve)){
    //when approve is pressed get all the drawn geometries apply monte carlo simulation
    //and add them to defined geometries panel
    try {
        double stdevX = Double.parseDouble(stdevXField.getText());
        double stdevY = Double.parseDouble(stdevYField.getText());
        int simulationCount = Integer.parseInt(simulationNumField.getText());
        confidence = Double.parseDouble(confidenceField.getText());
        simulator = new PBMMonteCarloSimulator(stdevX, stdevY, simulationCount);
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Enter valid numarical values for statistics.",
                "Unaccepted Statistics", JOptionPane.WARNING_MESSAGE);
    }
    constructGeometry(dataLine, LINE);
    constructGeometry(dataArc, ARC);
    constructGeometry(dataSpline, SPLINE);
    statTabs.setComponentAt(1, getDefinedGeometriesPanel());
    statTabs.repaint();
    statTabs.setSelectedIndex(1);
    dataLine = new Vector();
    dataArc = new Vector();
    dataSpline = new Vector();
    lastGeomId = geomID;
    repaintAllGeomTables();
    if(featureVector.size()!=0)
        saveWork.setEnabled(true);
    else
        saveWork.setEnabled(false);
}
```

C.2 Clear All Action

```
if(e.getSource().equals(clearAll)){
    geomHandler.removeGivenFeatures(dataLine, dataArc, dataSpline);
    dataLine = new Vector();
    dataArc = new Vector();
    dataSpline = new Vector();
    if(featureVector.size() != 0){
        IDFeatureHolder holder = featureVector.get(featureVector.size()-1);
        IPBMNetworkFeature f = holder.getOriginalFeature();
        geomID = f.getFeatureID();
    }else
        geomID = lastGeomId;
    repaintAllGeomTables();
    geomHandler.setGeomCounter(geomID);
    geomHandler.refresh();
}
```

C.3 Save Work Action

```
if(e.getSource().equals(buttonOK)){
    dialog.dispose();
    IDStatisticsFileGenerator fileGenerator = new
    IDStatisticsFileGenerator(cgfFileContent);
    String directory = fileSaveDirField.getText();
    String fileName = fileNameText.getText();

    if(directory.equals("") || directory == null){
        JOptionPane.showMessageDialog(dialog,
                "Select a suitable director to save the file.",
                "Warning-Missing Directory", JOptionPane.WARNING_MESSAGE);
    }else{
        if(fileName.equals("") || fileName == null)
            JOptionPane.showMessageDialog(dialog,
                    "Enter a suitable file name to save already done work.",
                    "Warning-Missing File Name", JOptionPane.WARNING_MESSAGE);
        else{
            fileGenerator.createFile(directory, fileName);
            fileGenerator.calculateStatistics(featureVector,
                    geomDataDefinedHash, confidence);
        }
    }
}
```

C.4 Monte Carlo Simulation

Approve action constructs Monte Carlo Simulator class **PBMMonteCarloSimulator** and given code block is used to simulate the given coordinate.

```
public GKCoordinate simulate(GKCoordinate baseCoord){
    try {
        long seedLong = sRand.nextLong();
        Random random = new Random(seedLong);
        double randX = random.nextGaussian();

        seedLong = sRand.nextLong();
        random = new Random(seedLong);
        double randY = random.nextGaussian();

        double newX = randX * stdevX + baseCoord.x;
        double newY = randY * stdevY + baseCoord.y;

        return new GKCoordinate(newX, newY);
    } catch (Exception e) {
        e.printStackTrace();
        return baseCoord;
    }

}
```

Figure C.1: PBMMonteCarloSimulator simulate function for a given coordinate

APPENDIX D

COMPARISON OF G-BAND AND PROPAGATED ERROR BAND

D.1 Comparison of Band Areas

Tables D.1 and D.2 give the calculated area of the generated band geometry for each line separately.

Table D.1: Generated Band Areas for Each Line

Line ID	*G-Band Area*	*Propagated Error Band Area (Line Based)*	*Propagated Error Band Area (Network Based)*
1	185.31	184.36	184.03
2	127.23	160.09	148.49
3	117.2	96.4	105.36
4	66.98	73.35	72.83
5	45.74	56.11	60.8
6	132.14	137.31	155.54
7	174.58	176.76	176.81
8	63.16	78.18	86.13
9	68.95	73.87	90.08
10	730.8	759.3	901.05
11	80.62	84.31	84.76
12	35.22	43.95	45.31
13	116.81	143.47	161.78
14	85.58	101.08	128.92
15	80.87	101.63	118.42
16	41.45	43.82	48.84
17	44.1	47.84	48.65
18	74.3	73.91	76.13
19	114.2	112.96	115.32
20	44.11	55.94	61.05
21	49.17	45.54	44.25

Table D.2: Generated Band Areas for Each Line Continue

Line ID	*G-Band Area*	*Propagated Error Band Area (Line Based)*	*Propagated Error Band Area (Network Based)*
22	54.26	49.46	47.26
23	78.59	85.4	93.8
24	108.33	120.79	132.84
25	48.52	52.19	58.81
26	50.57	55.93	51.41
27	41.69	41.46	42.63
28	60.13	66.7	67.43
29	186.26	219.36	228.38
30	75.01	79.2	82.8
31	61.13	79.7	84.71
32	304.43	288.15	389.9
33	95.88	109.96	110.53
34	81.05	100.49	108.55
35	228.25	229.92	255.25
36	122.95	141.65	167.77
37	170.95	243.6	268.94
38	190.46	208.86	257.44
39	96.6	90.19	88.34
40	58.1	58.87	72.65
41	29.2	31.13	31.53
42	48.14	48.96	48.96
43	95.71	84.89	80.57
44	67.72	58.66	54.49
45	90.55	87.07	87.68
46	34.4	43.43	48.46
47	60.46	54.19	61.71
48	39.48	42.84	48.81
49	213.11	207.77	213.15
50	82.84	78.92	72.18
51	73.17	74.94	75.07
52	318.59	351.9	345.29

Table D.3 gives the sum of the generated band areas. This sum is a line based sum. After overlaying the generated band geometry on the connection points of the network, this sum will decrease.

Table D.3: Generated Band Areas Total (Line Based Sum)

Band Type	*Total Area(Line Based Sum)*
G-Band	5745.33
Propagated Error Band(Line Based)	6137.05
Propagated Error Band(Network Based)	6692.14

D.2 Comparison of Base and Sample Geometry Coverage

Tables D.4 and D.5 give the number of sample lines covered by the band geometry. G-Band IN, Propagated Error Band Model (PBM)(Line) IN and PBM(Network) IN columns involve the number of sample lines which lie inside the generated G-Band, Propagated Error Band (Line Based) and Propagated Error Band (Network Based) geometries, respectively. Also G-Band OUT, Propagated Error Band Model (PBM)(Line) OUT and PBM(Network) OUT columns involve the number of sample lines which lie outside of G-Band, Propagated Error Band (Line Based) and Propagated Error Band (Network Based) geometries, respectively.

Table D.4: Base and Sample Geometry Coverage (Line Based)

Line	*G-Band IN*	*PBM(Line) IN*	*PBM(Network) IN*	*G-Band OUT*	*PBM(Line) OUT*	*PBM(Network) OUT*
1	12	11	11	3	4	4
2	9	12	11	6	3	4
3	10	10	11	5	5	4
4	12	12	12	3	3	3
5	8	10	11	7	5	4
6	12	12	9	3	3	6
7	9	9	9	6	6	6
8	10	11	12	5	4	3
9	12	13	13	3	2	2
10	10	10	12	5	5	3

Table D.5: Base and Sample Geometry Coverage (Line Based) Continue

Line	*G-Band IN*	*PBM(Line) IN*	*PBM(Network) IN*	*G-Band OUT*	*PBM(Line) OUT*	*PBM(Network) OUT*
11	12	13	13	3	2	2
12	11	12	14	4	3	1
13	11	12	13	4	3	2
14	10	10	11	5	5	4
15	10	12	13	5	3	2
16	9	9	12	6	6	3
17	11	12	12	4	3	3
18	9	9	10	6	6	5
19	11	10	10	4	5	5
20	9	10	10	6	5	5
21	10	9	10	5	6	5
22	11	9	7	4	6	8
23	10	10	9	5	5	6
24	11	12	12	4	3	3
25	11	11	11	4	4	4
26	8	9	9	7	6	6
27	10	10	11	5	5	4
28	11	13	13	4	2	2
29	10	10	10	5	5	5
30	8	9	7	7	6	8
31	11	12	13	4	3	2
32	10	7	7	5	8	8
33	7	7	7	8	8	8
34	11	12	12	4	3	3
35	12	12	12	3	3	3
36	11	11	12	4	4	3
37	9	11	11	6	4	4
38	9	8	11	6	7	4
39	11	11	11	4	4	4
40	11	6	7	4	9	8
41	8	8	8	7	7	7
42	8	8	8	7	7	7
43	7	12	12	8	3	3
44	11	9	8	4	6	7
45	11	11	10	4	4	5
46	10	10	11	5	5	4
47	11	11	12	4	4	3
48	14	12	13	1	3	2
49	13	11	11	2	4	4
50	11	11	11	4	4	4
51	10	10	10	5	5	5
52	12	12	11	3	3	4

Table D.6 involves total number of lines which is inside and outside the implemented band model geometry.

Table D.6: Base and Sample Geometry Coverage Total

Band Type	*Total Inside Band*	*Total Outside Band*	*Total Lines*
G-Band	535	245	780
PBM (Line Based)	543	237	780
PBM (Network Based)	556	224	780