Space domain based modeling and inversion techniques of gravity anomalies using variable density-depth models

A Thesis submitted to the University of Hyderabad for the Degree of

DOCTOR OF PHILOSOPHY

in University Centre for Earth and Space Sciences, School of Physics

by

M. Pramod Kumar

November, 2015

CERTIFICATE

This is to certify that the thesis entitled "*Space domain based modeling and inversion techniques of gravity anomalies using variable density-depth models*" submitted by Mr. M. Pramod Kumar, bearing regd. No 13ESPE02, in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in Geophysics is a bonafide work carried out by her under my supervision and guidance which is a plagiarism free thesis.

The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Date: 26.11.2015

(V. Chakravarthi)
Supervisor

Head of the Centre

Dean of the School

Declaration

University Centre for Earth and Space Sciences
School of Physics
University of Hyderabad
Hyderabad – 500 046

Dated: 26.11.2015

This is to certify that I, M. Pramod Kumar have carried out the research embodied in the present thesis for the full period prescribed under Ph.D ordinances of the University.

I declare to the best of my knowledge that no part of this thesis was earlier submitted for the award of research degree of any University.

(Signature of the candidate)
Name: M. Pramod Kumar
Enrollment No: 13ESPE02

Signature of the Supervisor

Head of the Department **Dean of the School**

To my beloved parents….

Acknowledgements

It is with a deep sense of indebtedness and gratitude that I thank my research supervisor Dr. V. Chakravarthi, Associate Professor, University Centre for Earth and Space Sciences (UCESS), University of Hyderabad for his expert guidance and constant encouragement throughout the progress of my research work. His ever friendly nature and obliging attitude made my every scientific discussion with him something special.

Special thanks go to the Doctoral Research Committee members, Professor B. Madhusudan Rao and Dr. S. Sri Lakshmi, for giving appropriate inputs, when and wherever required, during the course my research work.

I sincerely thank Professor M. Jayananda, Head, UCESS for providing all the necessary facilities to carry out my research work. Drs. P. S. Roy, K. Ashok, and Sri T. Suryanarayana have extended their timely support and encouragement, for which I am ever grateful.

I feel it as ethical in my part to express my sincere gratitude to University of Hyderabad for providing this opportunity to pursue my research work leading to the award of Ph.D Degree. University Grants Commission (UGC), New Delhi is profusely acknowledged for extending financial support in the form of Rajiv Gandhi National Fellowship.

Dr. B. Ramamma, Mr. K. Mallesh, Mr. N. Naveen Kumar have helped me a lot during the final compilation of my thesis. I would like to thank all of them including my fellow research scholars and

supporting staff in the Centre for their benevolent attitude and friendliness.

I owe my affection to my friends Rupini Priyanka, Neelesh Babu and Nagabhushanam with whom I have shared all my happiness and sorrows. Their association is indeed a fabulous gift given to me by the Almighty.

Words fall short to express my deep sense of respects and regards to my parents and other family members for their cordial support and encouragement all through my academic career, for which I am ever indebted.

***

Contents

Preface

Geophysical methods play an important and indispensable role in the exploration of natural mineral resources such as oil and natural gas, hydrogeological investigations, geodynamic and tectonic studies, engineering and environmental problems, issues related to glaciology and volcanology, archaeological investigations etc. A wide spectrum of geophysical methods is in vogue for the exploration of the subsurface depending on the nature and type of the problem under consideration. The contrasts in physical properties such as the density, magnetization, electrical resistivity/conductivity, elastic properties etc. between the target source and the surroundings pave the way for the measurement of the corresponding anomalous fields. Unwarranted geophysical signals are suppressed/removed from the measured ones before being subjected to analysis for the subsurface structures.

In this direction, it is emphasized in unequivocal terms that the gravity method, which is the subject of this thesis, depends exclusively on considerable density contrast between the source/sources of interest and the surrounding formations, which generates a favorable gravity anomaly. Although, the density of sedimentary rocks does not lend itself to be simulated by any mathematical formulation; sometimes it is possible to model the density variation of sediments with depth by an exponential law. However, such simulation by an exponential law makes it difficult to derive closed form analytical expressions for gravity anomalies in the space domain. On the other hand, closed form gravity expressions could be derivable in the frequency domain using exponential density-depth relationship; however, such forward modeling schemes find restricted practical application because truncation errors arise while transforming the anomalies from the frequency to the spatial domain. Thus it is imperative to develop appropriate mathematical tools in the space domain not only to realize forward gravity modeling of geologic sources/structures but also to estimate the source parameters making use of the exponential density function.

In the thesis, both analytical and numerical approaches have been used judiciously in the space domain to derive the expressions for calculating the gravity anomalies of selected geologic structures using Exponential Density Contrast Model (EDCM). Based on these forward modeling schemes, new automatic techniques are developed in the spatial domain using EDCM to interpret gravity anomalies due to 2D listric fault morphologies and sedimentary basins besides relevant GUI based softwares. Furthermore, a semiautomatic/interactive gravity modeling scheme and an optimization coupled with relevant softwares are developed to interpret the gravity anomalies produced by 2.5D listric fault sources, wherein the hanging wall systems consist of several formations. Applicability of each proposed technique is demonstrated both with synthetic and real field data analysis. The results are highly encouraging and thereby substantiating the validity of the proposed techniques.

The thesis under study is organized into seven chapters as detailed below.

In chapter-I, general introduction of the gravity method, mathematical foundations, concept and role of exponential density contrast model, and a brief account of the earlier work done on modeling and inversion techniques are discussed.

In Chapter II, the principles of inversion are used to develop an automatic technique in the space domain to interpret the gravity anomalies of 2D listric fault morphologies, wherein the density contrast within the detached hanging wall varies exponentially with depth. The fault ramp of the structure is described by a predefined polynomial function having arbitrary but specific degree, whose coefficients become the unknown parameters to be estimated from the observed gravity anomalies in addition to the thickness of the detached hanging wall. The proposed inversion identifies approximate parameters pertaining to the location of the origin of fault plane and depth to the decollement horizon from a set of characteristic anomalies. Based on the errors between the observed and modeled gravity anomalies, the inversion technique constructs and subsequently solves the system of normal equations to estimate the improvements in the depth and coefficients of the polynomial in an iterative approach until the specified convergence criteria is fulfilled. Based on the proposed inversion methodology, a GUI based software, GRIN2DFL, coded in JAVA is developed. This code, works on Model-View-Controller (MVC) pattern, reads the input parameters as specified by the interpreter and analyzes the anomalies for the concealed structure in an automatic manner. The software has inbuilt graphical user interface, which enables the interpreter to visualize the animated versions of the model growth and corresponding improvement of model response, changes in misfit, and variation of density contrast with depth. The efficiency of the inversion and software are illustrated with the gravity anomaly of a synthetic model of a 2D listric fault source in the presence of pseudorandom noise. The real field gravity anomalies across the Ahiri-Cherla master fault of the Godavari sub-basin in India are analyzed and found that the results are in line with the available/reported geologic information.

A semiautomatic/interactive modeling technique coupled with relevant GUI based JAVA code, FRGMLSTRK, is developed in chapter-III to analyze the gravity anomalies of 2.5D strike limited listric fault sources in real time. The hanging wall systems may consist in any number of formations irrespective of their thicknesses and densities. The fault ramps of the structures are described with analytical functions. The effects of profile azimuth and strike length of the structure on the magnitude of gravity anomaly due to a 2.5D listric fault source are demonstrated in length. The proposed scheme allows one to construct the fault plane geometry, depths and the densities of various subsurface formations in an interactive mode using simple mouse operations. The business logic of the algorithm computes the gravity response arises from the model in real time and the inbuilt graphical user interface compares the model response with the observed anomalies. The differences between the said two anomalies could be minimized by modifying the model space, density and depth parameters either independently or in combination. This could be realized by simple drag and drop mouse operations. The applicability of the method and software is exemplified with both synthetic and real field gravity anomalies. In case of field example, the analysis of gravity anomalies across the Aswaraopet master fault from the eastern margin of the Chintalpudi sub-basin in India has yielded a structure that is marginally deviated from the one realized by Deep Seismic Sounding (DSS) studies.

Chapter-IV deals with the development of an inversion technique and associated GUI software in JAVA, using the principles of inversion, to simultaneously estimate the geometry of fault ramps and density or thickness parameters of the formations within the hanging wall systems of strike limited listric fault sources. The proposed inversion requires initial/guess parameters pertaining to the densities and depths of the formations within the hanging wall systems, whereas the parameters required for initiating the fault plane are calculated automatically. The MVC based software, GRAVLIS, reads the input parameters specified by the interpreter and performs the inversion to recover the structure. In addition to generating output in ASCII and graphical forms, the software displays the animated versions of model space improvement and corresponding changes in model gravity response and density-depth improvement with the iteration number. The efficiency of inversion is demonstrated with a set of noisy gravity anomalies of a 2.5D synthetic model. The inversion when performed on the gravity anomalies of the Aswaraopet master fault of the Chintalpudi sub-basin has yielded results that are excellently comparable with the borehole information.

Based on the principles of modeling, an automatic technique coupled with GUI based software is developed using EDCM to model the gravity anomalies arise from 2D sedimentary basins and presented in chapter-V. The cross-section of a sedimentary basin is simulated with a polygonal source defined with appropriate number of vertices. The depth co-ordinates of the vertices of the polygon become the unknown parameters to be found from the observed gravity anomalies. Expression for the gravity anomaly of such a source with EDCM is derived in the space domain by making use of the Stoke's theorem. The validity of the proposed numerical method of forward modeling is demonstrated on three regular geophysical models, namely; a prism, vertical fault and a trapezoid. In all the cases, the anomalies realized from the proposed numerical method compare excellently well with those obtained from respective analytical gravity expressions of the models. The present modeling method computes the initial depths of a sedimentary basin from the observed gravity anomalies and updates them automatically in an iterative approach within the specified convergence criteria. Based on the algorithm, a software, MOD2DGREXP, coded in JAVA, with GUI compatibility is developed. This code, works on MVC pattern, reads the residual gravity anomalies of a sedimentary basin and estimates basement depths at plurality of locations on the profile. Besides generating the output in both ASCII and graphical forms, the software displays the changes in the depth structure, nature of fit between the observed and modeled gravity anomalies, changes in misfit between the observed and model gravity anomalies, and variation of density contrast with depth against iteration in animated forms. The efficiency of proposed modeling is illustrated with a synthetic model of a sedimentary basin, whose anomalies are corrupted with random noise. In case of real field example, the gravity anomalies of the San Jacinto graben, California are analyzed using a derived exponential density contrast model. The estimated depth structure is compared with the one previously reported along with the available subsurface geologic information derived from seismic studies.

In Chapter-VI, an automatic gravity optimization technique, using the principles of inversion, is developed together with a GUI based software to estimate the depths to the basement topography above which the density contrast follows exponential decay with depth. The density interface between the sediments and underlying basement is simulated with a polygonal geometry with suitable number of vertices. Unlike the case with automatic modeling, this technique make use of the errors between the observed and model anomalies at all the locations to build the system of normal equations. These equations are solved for the improvements in the depth ordinates of the polygon. Depth parameters of the vertices are updated and the process repeats till the convergence criterion is fulfilled. The software, IN2DGREXP, reads the input parameters and performs the business logic of the inversion to estimate the depths of the density interfaces in an automatic mode. The GUI capability of the software enables the interpreter to visualize the convergence of the solution with the iteration. The inversion when performed on the two gravity anomalies considered in the previous Chapter-V has yielded structural solutions that are more or less coincide with those obtained from automatic modeling. It was found that the inversion performs lesser number of iterations for proper convergence of the solution in comparison to automatic modeling.

All the softwares' presented in this thesis are platform independent.

Chapter-VII summarizes the overall conclusions from the thesis and scope for future research.

***

List of Figures

List of Tables

List of research publications (SCI)

Chakravarthi, V., Rajeswara Sastry, S., **Pramod Kumar, M.**, 2014, A method and a GUI based JAVA code for interactive gravity modeling of strike limited listric fault sources with arbitrary density-depth variations, Journal of the Geological Society of India (Springer), 83, 577-585.

Chakravarthi, V., **Pramod Kumar, M.**, 2015, Estimation of multiple density-depth parameters from gravity inversion: Application to detached hanging wall systems of strike limited listric fault morphologies, Geophysica Internacional (Springer), 54, 49-65.

Chakravarthi, V., **Pramod Kumar, M.**, Ramamma, B., Rajeswara Sastry, S., 2015, Automatic gravity modeling of sedimentary basins by means of polygonal source geometry and exponential density contrast variation: Two space domain based algorithms, Journal of Applied Geophysics (Elsevier), DOI: 10.1016/ j.jappgeo.2015.11.007.

Chakravarthi, V., **Pramod Kumar, M.**, Ramamma, B., Rajeswara Sastry, S., 2015, Gravity anomaly interpretation of 2D listric fault morphologies using exponential density contrast model: A space domain technique, Under review with Arabian Journal of Geosciences (Springer).

CHAPTER

ONE

Introduction

1.1 General

Geophysical methods are the inevitable tools for the investigation of subsurface geology at micro, macro and global scales. In addition, these methods occupy a strategic role in the exploration programs for natural resources. Several geophysical methods viz., gravity, magnetic, electrical, electromagnetic, seismic, radiometric etc. are in vogue to map and explore the concealed geology. Each geophysical method exploits the corresponding physical property contrast to produce relevant geophysical signals/anomalies for exploring the subsurface. These geophysical signatures/anomalies shall be collected on the ground surface, boreholes and also from airborne platforms and subsequently processed and analyzed in terms of the parameters to quantify the concealed geology in detail.

However, in deciding which geophysical method or combination of methods to use under a given set of circumstances, close attention must be given to the inherent suitability of each method for investigating the problem under consideration. It may not be feasible to use the most suitable one and, as a matter of expediency, a second or third choice procedure may also have to be adopted.

Undoubtedly, sophisticated and state-of-the-art instruments and computer aided technology made the geophysical field data acquisition, processing and interpretation effective and elegant, but at times, the process of interpretation continues to be cumbersome because of the subsurface heterogeneity and complexity. Although, interpretation techniques based on the thumb rules, curve matching etc. are in vogue; the development of highly powerful algorithms almost surpass the existing old techniques.

Interpretation of geophysical data involves two aspects; the first one is to estimate the parameters of the causative body, which is responsible for generating the anomalous field and the second one is to translate the interpreted geophysical model in terms of concealed geology. However, quantitative interpretation of field measurements taken at the surface is not unique because of the fact that the observed anomalies on the topography are the cumulative effect of useful signal and noises of different origin. Moreover, the observed anomaly on the topography can be equally explained by a number of sources. Such ambiguity and uncertainty in the interpretation can be reduced to a large extent by incorporating additional source of information obtained from surface outcrops, bore holes, and by other geophysical methods.

Gravity exploration, which is the subject matter of the thesis, comes under the category of passive geophysical methods. It does not rely on controlled sources but seeks out naturally occurring variations in the earth's gravity field. A body rotating with the earth experiences the gravitational force of the masses of the earth, other celestial bodies, as well as the centrifugal force due to the earth's rotation. If the effects of centrifugal force and that of celestial bodies are removed the left out component becomes gravitation. The force of attraction between two point masses m_1 and m_2 is directly proportional to the product of their masses and inversely to the square of the distance between the centers of respective masses and can be expressed as

$$F = \frac{-Gm_1m_2}{r^2}\frac{k}{r}, \tag{1.1}$$

where, F is the force on m_2, r is the distance between m_1 and m_2 and G is the universal gravitational constant. The minus sign indicates that the force is always attractive. The vector k may be expressed by the position vectors r and r' in the Cartesian coordinate system as

$$k = r - r', r^T = (X, Y, Z), r'^T = (X', Y', Z'), \tag{1.2}$$

with the magnitude of

$$k = \sqrt{(X - X')^2 + (Y - Y')^2 + (Z - Z')^2}. \tag{1.3}$$

By setting the mass at the attracted point to unity, the above equation transforms into the gravitational acceleration represented by

$$g = \frac{-Gm}{r^2}\frac{k}{r}. \tag{1.4}$$

The earth, however, is composed of an infinite number of differential mass elements, dm. The gravitation on the unit mass then results from the integral over the individual contributions. The equation for gravitational acceleration then takes the form (Moritz, 1980; Chakravarthi, 2011a)

$$g = -G \iiint \frac{r - r'}{|r - r'|^3} dm. \tag{1.5}$$

The mass element dm can be expressed as

$$dm = \sigma du, \tag{1.6}$$

where, σ is the density of the volume element, du. Because the gravitational field is invariant to rotations

$$curl\ g = 0. \tag{1.7}$$

Also, the vector g may be expressed as the negative gradient of the potential U (Sigl, 1985),

$$g = -grad\ U. \tag{1.8}$$

For a point mass m, the gravity potential can be expressed as

$$U = \frac{GM}{r}, with \lim_{r\to\infty} U. \tag{1.9}$$

For the earth,

$$U = G \iiint \frac{dm}{r} = G \iiint \frac{\sigma}{r} du, \lim_{r\to\infty} U. \tag{1.10}$$

From the above relations, it is clear that once the density function σ is known for the earth, the gravitation can be calculated as a function of the position.

The acceleration of gravity in the direction of z-axis can be derived from equation (1.9) with reference to the origin of the Cartesian coordinate system as

$$g_z = -\frac{\partial U}{\partial z} = G\sigma \iiint \frac{z}{r^3}\,dxdydz, \tag{1.11}$$

where, $r^2 = x^2 + y^2 + z^2$.

If the body is very long in one direction, say, along the y-axis and has an uniform cross-section in the xz-plane, the gravity attraction can be obtained from a logarithmic 2D potential as

$$U = 2G\sigma \iint \log\left(\frac{1}{r}\right) dxdz\,. \tag{1.12}$$

Here, $dxdz$ is the cross-sectional area of a 2D element within the body. The gravity effect of the 2D body can be expressed as

$$g_z = -\frac{\partial U}{\partial z} = 2G\sigma \iint \frac{zdxdz}{r^2}, \tag{1.13}$$

where, $r^2 = x^2 + z^2$.

1.2 Exponential Density Contrast Model (EDM)

Equation (1.11) and (1.13) can be used to calculate the gravity effects of 3D and 2D geologic sources under the assumption that these sources possess uniform density throughout. Unequivocally, the success of the gravity method is largely dependent on the existence of significant lateral density contrast between the source/sources of interest with the adjacent formations. The dimensions of the source and its depth of occurrence also influence the magnitude of gravity anomaly on the topography. Hence, the knowledge of the density of rocks both at the surface and subsurface is always essential not only for applying a few gravity corrections to the gravity measurements but also to obtain reliable geologic interpretations of the gravity anomalies.

In case of sedimentary rocks the density is rarely uniform (see for e.g., Athy, 1930; Hamilton and Menard, 1956; Becking and Moore, 1959; Prozorovich, 1960; Gealy, 1969; Hamilton, 1976; Bachman and Hamilton, 1976; Helmberger et al., 1979; Donato and Tully, 1981; Dimitropoulos and Donato, 1981; Foucher et al., 1982; Zervos 1987; Holliger and Klemperer, 1989; Thorne and Watts, 1989; Artemjev et al., 1994; Angell et al., 1997; Chaika and Williams, 2001; Azeglio et al., 2010; Wang et al., 2011; Gu et al., 2014; Martinez et al., 2014).

Athy (1930) was the first researcher to express the relations between the depth of burial and the density, porosity, and compaction of different types of sediment by exponential equations. Manger (1963) had documented large amount of measured data pertaining to the porosity and density measurements of sedimentary rocks and concluded that the porosity of sandstones generally decreases whereas the density increases with depth of burial and age. From an extensive study and analysis of the density logs measured in 435 deep wells in Western Canada, Maxant (1980) had shown that the density-depth relationship seldom follows a linear trend; and in the case of shale, the correlation between density and depth could be explained by exponential density-depth relationship. Dickinson (1953), Dallmus (1958), Storer (1959), McCulloh (1967), Eaton

(1969), Rieke et al. (1974), Castagna et al. (1993) have demonstrated that the measured densities for shale as a function of depth showed more or less similar behavior, although the samples were collected from a wide variety of locations with different geologic settings and histories.

On the other hand, Cowie and Karner (1990) have demonstrated from the measured density-depth data of different stratigraphic units in many sedimentary basins that the sediment densities exhibit a wide range but the mean density clearly increases with depth with the highest rate in the top few hundred meters. They have also showed that the density of sediments near the depocentres approaches to that of the basement rocks whereas near the basin margins, the sediments portray relatively lower density.

Recently, Tenzer and Gladkikh (2014), based on the analysis of density samples taken from 716 drill sites of the Deep Sea Drilling Project (DSDP), showed that the density increases nonlinearly with the increasing sediment depth due to compaction. They have also proposed theoretical models to compute the density contrasts of the interfaces between ocean-sediment and sediment-bedrock.

Cordell (1973), Cai and Zhdanov (2015) argue that the density-depth relationship of sedimentary rocks, in general, does not strictly follows any mathematical formulation because of the influence of several geologic factors such as compaction, stratigraphic layering, cementation, facies change, diagenesis etc. Cordell (1973) had established from the actual measurement of sedimentary rock density from deep bore holes that the density contrast decreases drastically at shallow depths and less progressively at deeper depths. Such variation in density contrast of sedimentary rocks with depth could be effectively modeled by an exponential law (Cordell, 1973; Granser, 1987; Chai and Hinze, 1988; Chappel and Kusznir, 2008; Chakravarthi et al., 2015a, 2015c). Hence, the use of Exponential Density Contrast Model (EDCM) in the analysis of gravity anomalies ensures more reliable interpretations.

The exponential density contrast model is mathematically defined as (Cordell, 1973)

$$\Delta\rho(z) = \Delta\rho_0 e^{-\lambda z}, \tag{1.14}$$

where, $\Delta\rho(z)$ is the density contrast of the sediments at a given depth, z, $\Delta\rho_0$ is the density contrast at the surface/topography and λ is a decay factor expressed in reciprocal length units. The values of $\Delta\rho_0$ and λ can be obtained by fitting equation (1.14) to the known density contrast-depth data of subsurface geology.

However, the major intricacy associated with the exponential density contrast model is that analytical expressions for the gravity anomalies could not be derivable in the space domain even for simple geophysical models (Chakravarthi and Sundararajan, 2004). Although, closed form solutions are possible in the frequency domain with this density contrast model to realize forward modeling (Granser, 1987; Chai and Hinze, 1988; Chappel and Kusznir, 2008), significant truncation errors would popup in the analysis when the anomalies transform back to the space domain (Chakravarthi and Sundararajan, 2007).

In this thesis, new approaches have been formulated in the spatial domain to calculate the gravity anomalies of geologic structures using an exponential density contrast model by judiciously combining both analytical and numerical approaches. New interpretation techniques with relevant softwares are then developed to recover the structures from the observed gravity anomalies. From the known density-depth information, the Exponential Density Contrast Model (EDCM) is constructed for the field data presented in chapters II, V and VI.

1.3 Gravity corrections

The end product of a gravity survey yields a set of gravimeter readings and a set of station elevations measured with reference to the mean sea level or geoidal surface. Absolute gravity is established at each observation by connecting the field gravity meter readings to a base station. These absolute gravity values represent the cumulative effects of the sources being looked for, in addition to the earth's topography, its non-sphericity of shape and its centrifugal force. In principle, normal gravity is calculated on the spheroidal surface of the earth, projecting it to the level of observation and adding to it the gravity effect of the masses between the spheroid and the ground surface. This can be realized by applying a series of corrections viz., normal or latitude correction, free-air correction, Bouguer correction and topographic correction to the measured gravity data. A detailed account of the application of gravity corrections is given by Rao and Murthy (1978). When the elevation differences between the field stations are small, the anomalies are deemed to have been made on a mean horizontal plane coinciding with the topography. Finally, the Bouguer gravity anomalies are presented in the form of contour maps and/ stacked profiles before being subjected to interpretation.

1.4 Interpretation

Interpretation of gravity anomalies is generally carried out in a four-prong strategy viz., i) qualitative interpretation of Bouguer gravity anomaly maps, ii) regional and residual anomaly separation, which is often supplemented with derivative calculations and continuation, iii) quantitative interpretation, and iv) geological translation of geophysical interpretations.

1.4.1 Qualitative interpretation

In qualitative interpretation, gravity highs and lows are identified on a contour map and their axes are correlated with the known surface or subsurface geology and the correlations are extrapolated to poorly mapped areas. Further, the nature and characteristics of the anomalies are studied to identify the source geometry, its orientation besides its approximate depth of occurrence.

Two-dimensional bodies of large strike lengths are represented by anomaly contours elongated roughly parallel to the strike of anomalous bodies; conversely the direction of elongation of contours or their axis is a measure of the strike of the anomalous body. Three-dimensional bodies produce elliptical or closed contours.

When the anomalies are presented in the form of profiles, typical characteristics of individual profiles are being identified and correlated with those on neighboring profiles. Characteristic features persist on neighboring profiles indicate the strike/orientation of the anomalous body. Fault structures create dislocation or complete absence of characteristics on neighboring profiles. Widths of individual anomalies and their sharpness decide the order of depth or horizontal dimensions of the body.

1.4.2 Regional and residual anomaly separation

The measured Bouguer gravity anomalies on the topography are the cumulative gravity effects of the sources distributed both laterally and vertically. The contribution of gravity field due to the source/sources of interest is known as the residual gravity anomalies, which are high frequency in nature. The sources located at deeper levels/far away from the source/sources produce low frequency gravity signals, known as regional gravity anomalies. The gravity field due to sources above the source of interest is known as noise and can be identified and removed from the Bouguer anomalies owing to its conspicuous and random nature of occurrence. The process of separating the regional gravity anomalies from the residuals is known as the regional and residual anomaly separation.

Among many others, the methods based the graphical and smoothing techniques, empirical gridding, second derivative etc. (Telford et al., 1990), upward continuation (Jacobsen, 1987; Pacino and Introcaso, 1988), minimum curvature (Mickus et al., 1991), orthogonal polynomial fitting (Grant, 1957; Forsythe, 1957; Spitz, 1966; Merriam and Cocke, 1967; Beltrao et al., 1991; Agarwal and Sivagi, 1992), linear filtering (Strakhov, 1964; Strakhov and Lapina, 1967, Naidu, 1966; Naidu, 1967; Spector and Grant, 1970; Byerly, 1965; Sax, 1966; Lavin and Devane, 1970), nonlinear filtering (Naudy and Dreyer, 1968), Band pass filtering (Blakely, 1995), Wiener filtering (Pawlowski and Hansen, 1990), Green's equivalent layer (Pawlowski, 1994), fractals (Chapin, 1996), cellular neural networks (Albora et al., 2001); finite elements (Mallick and Sharma, 1999; Agarwal and Shalivahan, 2010), Hartley transform (Kadirov, 2000), multiscale edge and iterative lateral continuation and subtraction analysis (Boschetti et al., 2004), eigenimage extraction (Ganguly and Dimri, 2013), and higher order polynomial fitting (Chakravarthi et al., 2013a) have gained importance for separating regional gravity anomaly from the Bouguer anomalies. Nevertheless, known geology plays a decisive role in deciphering the best regional trend on a Bouguer anomaly map. The residual anomalies separated from the Bouguer gravity anomalies are subjected for the quantitative analysis.

1.4.3 Quantitative interpretation

Quantitative interpretation of gravity anomalies aims at estimating the parameters of geophysical models, which in turn explain the geology.

However, the interpretation of gravity anomalies is non-unique and ill-posed in the sense that the measured gravity anomalies on the plane of observation can be explained by a variety of density distributions as explained by Roy (1962), Backus and Gilbert (1967, 1968), and Blakely (1995).

To transform an ill-posed problem into a well-posed one, one may choose either to reduce the information demanded or introduce a *priori* information in the modeling space. Parker (1974,1975), Huestis and Parker (1977), Ander and Huestis (1987) have adopted the first approach to obtain the estimates for the lower and upper bounds of the physical property, the depth to the top and the thickness of the source. In contrast, the second approach of introducing a *priori* information could be followed provided the information were properly translated from the geological setting (Silva et al., 2001).

To introduce a *priori* information, one needs to consider an interpretation model either in physical property mode or geometric mode. In the former case, the subsurface containing the anomalous mass shall be viewed as an ensemble of elementary cells of fixed size and the density of each cell is estimated (Li and Oldenburg, 1998). In the later case, the density/density contrast of the anomalous body shall be specified/formulated and nonlinear operators are designed to determine the geometry of the source of the anomaly (Tarantola, 2005). Among the two approaches, the geometric mode is popular and being adopted by many researchers to develop interpretation algorithms to analyze the gravity anomalies (Murthy, 1998; Chakravarthi, 2003; Chakravarthi et al., 2013b, 2014, 2015a, 2015b, 2015c). Using the information derived from drilling/other geophysical data in the model space would further reduce the degree of uncertainty in the interpretation.

The nature and characteristics of the gravity anomalies very often reveal whether the anomalous source is a 2D, 2.5D and/or 3D structure (Rao and Murthy, 1978; Murthy, 1998). Suitable interpretation strategies can therefore be applied to interpret the gravity data.

1.5 Review of existing methods

Inverse problems are an important area of geophysical research because one has to make the quantitative estimates about the subsurface of the earth (i.e. to determine the unknown model space parameters) from a known/ measured set of observations on the surface of the earth.

The process of quantitative interpretation involves three important aspects namely, i) parameterization of model space, ii) forward modeling, and iii) optimization or inversion. A set of appropriate size and shape factors describe the model space. Physical laws governing the model space supplemented with relevant mathematics provide a means to compute the model gravity response. The optimization/inversion process recovers the unknown model space from the observed gravity anomalies.

Several techniques are being developed to analyze the gravity anomalies for subsurface structures in both spatial and wave number domains. In classical trial and error methods, approximate model space is constructed from a set of initial/guess parameters and the corresponding gravity response shall be calculated. Corrections are applied to the model parameters to minimize the data misfits between the observed and model/theoretical anomalies. This approach is repeated with several possible combinations of shape and size parameters till a satisfactory match between the observed and computed response is realized. Such an exercise is attempted with fast computing machines by means of suitable codes providing a comfortable graphical user interface and enabling a fast and easy way of changing the parameters of model space (Chakravarthi et al., 2014).

In wave number domain methods, Odegard and Berg (1965), Sharma and Geldart (1968), Collins et al. (1974), Mohan (1978), Bhimasankaram et al. (1977), Murthy and Rao (1980), Chacko and Battacharya (1980), Mareschal (1985), Rao et al. (1993), Annecchione et al. (2001) have proposed techniques based on the Fourier transforms, Sundararajan et al. (1983) have used the Hilbert transform, Shaw and Agarwal (1990) have applied Walsh transforms, Sundararajan and Brahmam (1998) have adopted the Hartley transform to interpret the gravity anomalies. Sundararajan et al., (2000), Al-Garni et al. (2010) have proposed techniques based on the Sundararajan transform to analyze the potential field anomalies of simple geophysical models. Generally, in spectral methods the gravity response of a source is analyzed within a moving window of predetermined size. The size of the window shall be determined based on an expected depth to the source, which is generally unavailable in advance (Chávez et al., 1999). Furthermore, significant errors would result in depth estimates in case the spectrum becomes complicated (Odegard, 2011).

On the other hand, the conventional Euler deconvolution method (Hood, 1965; Thompson, 1982; Wilsher, 1987, Corner and Wilsher, 1989; Reid et al., 1990; Klingele et al., 1991; Marson and Klingele, 1993; Fairhead et al., 1994; Huang et al., 1995; Reid, 1997; Zhang, 2000; Hu et al., 2011) is being widely used now-a-days to estimate source depths in a fast manner. However, this method invariably requires the information of the Source Structure Index (SSI). In order to choose SSI, the source needs to be approximated by specific geometries such as sphere, cylinder, etc. (Lafehr and Nabighian, 2012). In addition, wrong choice of SSI severely affects the depth estimates of the anomalous sources. Although, an extended Euler deconvolution (Mushayandebvu et al., 2001) can be used to simultaneously estimate the source depth and the structure index, this method is difficult adopt to deal with complex structures (Lafehr and Nabighian, 2012). Above all, the Euler deconvolution and spectral methods do not provide a direct comparison between the observed and modeled gravity anomalies; hence it becomes difficult to appraise the correctness of the solution.

The calculus based interpretational techniques such as the Newton and gradient methods solve the gravity inverse problems analytically if the functions describing the forward problem are differentiable (Tarantola, 2005). These methods are generally suitable if the objective function has one local minimum.

In case, the objective function is associated with several local minima then the global search optimization methods are preferred. These methods are categorized into two groups namely, guided and non-guided. The neural networks and evolutionary algorithms come under the guided methods, whereas the Monte Carlo methods fall under the category of non-guided methods. The neural networks operate on the basis of neurons to self organize and learn in respect of given external parameters (Brown and Poulton, 1996; Guang et al., 1998; Baan and Jutten, 2000; Osman et al., 2006, 2007; Leite and Filho, 2009). The evolutionary algorithms use the known principles of biological evolution to explore the whole parameter space for the best possible solution for a given inverse problem (Boschetti et al., 1997; Roy et al., 2002; Yao et al., 2003). The Monte Carlo methods search the model space randomly for the optimum solutions. These methods are very well suited if the number of unknown parameters to be solved are limited in number (Mosegaard and Tarantola, 1995; Tarantola, 2005).

Techniques based on wavelet transforms (Marlet et al., 2001; Hu et al., 2011; Oruç, 2014), 2D binary grids (Krahenbuhl and Li, 2006), tree-based geometry representation (Wildman and Gazonas, 2009), analytic signals (Beiki, 2010), Eigenvectors (Beiki and Pedersen, 2010) are also available to interpret the gravity anomalies. However, the approximation of uniform density for the anomalous source/sources in the enlisted methods is seldom valid particularly when reference is made to the structures associated with the sedimentary rocks.

In recent past, techniques based on tunneling algorithm (Levy and Montalvo, 1985, Mohan et al., 1986, Moharir, 1990); simulated annealing (Mundim et al., 1998; Nagihara and Hall, 2001; Roy et al., 2002; De Vicente et al., 2003; Jingxin et al., 2013), ant colony optimization (Dorigo and Blum, 2005; Gupta et al., 2011; Srivastava et al., 2013), particle swarm optimization (Kennedy and Eberhart, 1995; Eberhart and Kennedy, 1995; Shalivahan and Agarwal, 2010; Toushmalani, 2013; Pallero et al., 2015) are also found in literature. These strategies overcome the limitations associated with local optimization techniques.

Simulated Annealing (SA) simulates annealing process in which a substance is heated above its melting temperature and then gradually cools to produce the crystalline lattice, which minimizes its energy probability distribution. Ant Colony Optimization (ACO) is another evolutionary optimization algorithm which is inspired by the pheromone trail laying behavior of real ant colonies. On the other hand, Particle Swarm Optimization (PSO) is a population based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling searching for food. PSO shares many similarities with genetic algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However; unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. These algorithms are increasingly finding application in geophysical problems including potential field inverse problems, particularly in simultaneous inversion of gravity and magnetic data in order to reconstruct the shape of buried geological bodies; however, they are far more expensive both in terms of computational time and memory requirements.

1.6 Aim and scope of the thesis

The aim and objective of the thesis are to investigate the application of ridge regression algorithm in the inversion of gravity anomalies due to 2D and 2.5D listric fault morphologies, 2D sedimentary basins, besides developing automatic and semiautomatic modeling schemes coupled with relevant GUI based softwares coded in JAVA. Accordingly, the thesis under study is organized into seven chapters as detailed here under.

Chapter-I deals with general introduction, a brief account of various corrections to the measured gravity data, general principles of qualitative interpretation, concept of exponential density contrast model and a brief review of the earlier work on the development of gravity interpretation techniques etc.

In Chapter-II, an automatic inversion technique and related GUI based software to analyze the gravity anomalies of 2D listric fault morphologies with Exponential Density Contrast Model (EDCM) are dealt with. The reliability and applicability of the proposed technique and software are demonstrated on a synthetic model and also substantiated with real field data pertaining to the Ahiri-Cherla master fault of the Godavari sub-basin, India.

A semiautomatic/interactive modeling technique of gravity anomalies due to 2.5D strike limited listric fault sources with analytically defined fault planes and arbitrarily varying density-depth relationship is dealt with in Chapter-III along with relevant software. The effects of strike length of the structure and offset of the profile on the magnitude of gravity anomaly are described in detail. Both with a synthetic model of a 2.5D listric fault source and real field data measured from the eastern margin of the Chintalpudi sub-basin in India, the technique is demonstrated.

Chapter-IV deals with the development of an inversion technique and associated software in JAVA to simultaneously estimate the geometries of non-planar fault ramps and the parameters pertaining to either densities or thicknesses of formations within the detached hanging wall systems of 2.5D listric fault sources. Anomalies attributable to a synthetic model of a 2.5D listric fault source in the presence of pseudorandom noise and those observed across the Aswaraopet master fault of the Chintalpudi sub-basin, India are interpreted using the proposed inversion to demonstrate its applicability.

In Chapter-V, an automatic interpretation algorithm to analyze the gravity anomalies of 2D sedimentary basins with exponential density contrast model is dealt with along with relevant software programmed in JAVA. The algorithm and software operate on the principles of automatic modeling. The sediment basement interface is described with polygonal source geometry. The analysis is supported by a synthetic model and real field data across the San Jacinto graben, California.

In Chapter-VI, the principles of inversion are used to formulate an automatic optimization scheme and associated software in JAVA to estimate the depths of 2D density interfaces from the observed gravity anomalies. The density contrast within the sedimentary load follows exponential decay with depth. The successful interpretation of two gravity anomaly profiles, one synthetic and the other real, testifies the applicability of the proposed inversion.

In all the synthetic models presented in chapters II to VI the interpretations are compared with the assumed parameters, and in case of field data analysis the interpreted results are judged against the existing borehole /available geologic information.

A comprehensive conclusion of the entire work presented in the thesis is enumerated in Chapter-VII along with the scope for future research.

CHAPTER

TWO

Analysis of gravity anomalies of 2D listric fault morphologies using a prescribed exponential density contrast model*

2.1 General:

The primary goal of studying detailed gravity data is to provide better understanding of the subsurface geology. The gravity anomaly across a fault increases progressively to a maximum value over the uplifted side and a low over the downthrown block because the displacement of material causes a horizontal density contrast across the fault plane.

More often than not, the crustal extension is often acclimatized by high-angle faults that become almost listric at depth. These listric faults are curved normal faults in which the fault surface is concave upwards because the main detachment fracture follows a curved path rather than a planar path. The study of listric fault geometries is important because the movement along this type of fault is instrumental in forming important structural traps for oil and gas such as rollover anticlines and upthrown-fault-block closures etc.

---- ---

** Under review with Arabian Journal of Geosciences (Springer)*

Evidences for listric fault morphologies on the continental extensional regimes are plenty, to name a few, see for e.g., the Welshman's Rock, eastern Rhum (Emeleus, 1981), the North Sea Basin (Gibbs, 1983), the Corsair fault of offshore Texas (Christiansen, 1983), the Basin and Range (Wernicke and Burchfiel, 1982; Smith and Bruhn, 1984), the Murre fault of offshore Newfoundland (Tankard and Welsink, 1987), the Cordilleran fold and thrust belt (Constenius, 1996), the Cascadia continental margin (McNeill et al., 1997), Eskimo Lakes Fault Zone (Goussav et al., 2006).

Surface geologic studies do not easily reveal the listric nature of the faults because many a times the outcrop conditions prevent adequate geometric control of the fault planes. On the other hand, step like gravity anomalies can be observable if the detached rock masses on either side of such fault planes could create measurable lateral contrasts in rock densities. These gravity anomalies can be analyzed to estimate the parameters of such fault sources after properly accounting for regional gravity background.

Gravity anomalies due to two-dimensional bodies are elongated in one horizontal direction so that the anomaly length in this direction is at least twice the anomaly width. Such anomalies may be interpreted in terms of 2D structures, which theoretically extend to infinity in the elongate direction (strike) by using profiles at right angles to the strike direction (Kearey et al., 2002; Chakravarthi et al., 2015a).

2.2 Status of existing interpretational techniques

Many techniques are in vogue to analyze the gravity anomalies of fault structures. For e.g., Geldart et al. (1966) had proposed a curve matching technique, whereas Paul et al. (1966) demonstrated the use of the upward continuation to estimate the parameters of a fault structure. Chacko and Bhattacharya (1980), Murthy and Rao (1980), Pal (1981) have developed techniques based on the Fourier Transform, Sundararajan et al. (1983) and Mohan et al. (1986) had proposed schemes using the Hilbert transform and the Mellin Transform respectively to analyze the gravity anomalies of faulted beds.

Based on a least squares minimization approach Gupta (1983) had determined the depth of a buried faulted structure from the observed gravity anomalies. Thanassoulas et al. (1987) proposed a method and computer program to estimate the parameters of faulted beds. Abdelrahman et al. (1989), Gupta and Pokhriyal (1990) have developed methods using the amplitudes of maximum positive and negative gravity anomalies to determine the dip of the fault planes. Murthy and Krishnamacharyulu (1990) used the Marquardt algorithm to analyze the gravity anomalies of fault structures. McGrath (1991) had proposed a method based on lateral offsets of the zero-crossover point of the second horizontal derivative of an upward-continued gravity profile to estimate the dip, vertical extent, and location of the model boundary. Rao et al. (2003) used generalized inversion and single value decomposition techniques to model the gravity anomalies. On the other hand, Abdelrahman et al. (2003) have developed two least-squares approaches to determine the depth and amplitude coefficient of a buried faulted thin slab, successively from numerical first, second, third, and fourth derivative anomalies obtained from the observed gravity data using filters of successive graticule spacings. Stavrev and Reid (2010) used the concept of extended Euler homogeneity of potential fields to analyze the gravity anomalies of a faulted slab with large thickness relative to its depth. Aydogan (2011) had developed a technique based on the convolution between the templates obtained from the second horizontal derivative of the theoretical anomaly due to a truncated horizontal plate and the gravity anomaly to locate the positions of vertical or near vertical faults. Abdelrahman et al. (2013) developed a semi-automatic least squares method that uses first moving average residual gravity anomalies using filters of successive window lengths, whereas Essa (2013) proposed a technique using variance analysis to estimate the depth and dip angle of a buried fault structure. In recent past, Abdelrahman and Essa (2015) have developed three different least squares minimization approaches to analyze the gravity anomalies of dipping fault structures.

The practical utility of all the enlisted techniques becomes limited to analyze the gravity anomalies of the boundary faults associated with sedimentary basins because i) these faults are strongly curved in cross-section (Jackson and McKenzie,1983; Gans et al., 1985; Jackson, 1987; Janecke et al., 1998; Brady et al,. 2000; McKenzie and Jackson, 2012; Chakravarthi et al., 2015a), and ii) the density of sedimentary rocks is seldom uniform but increases with depth rapidly at shallow depths and less rapidly at progressively greater depths.

In this direction, although a few density functions were proposed and used by researchers in their interpretation strategies to analyze the gravity anomalies of fault structures viz., quadratic (Rao, 1985), linear (Sundararajan and Brahmam, 1998); these methods also presume planar surfaces for the fault planes. Moreover the linear density function, which is suitable to simulate the sediment density at large depths fails to explain the density variation at shallow depths (Chakravarthi and Sundararajan, 2007), whereas the quadratic density function falls short to replicate the true sediment density at depths as demonstrated by Chakravarthi (2009).

It is also to be realized that the use of existing commercial software, such as GM-SYS (Northwest Geophysical Associates Inc. 2004), to model the gravity anomalies of listric fault geometries is also problematic because a large number of constant density bodies are required to adequately explain the exponential density structure within the hanging wall (Zhou, 2013; Chakravarthi et al., 2015a).

For the above said reasons it is necessary to develop a new inversion technique coupled with software, which could overcome the drawbacks associated with the existing algorithms described in the text.

In this Chapter, two schemes are developed; one for realizing forward gravity modeling of a listric fault source from a known set of size and shape parameters and the other to perform automatic inversion on the observed gravity anomalies to recover the model space. In the later case, the unknown parameters to be estimated to quantify a listric fault source are: i) depths to the top and bottom surfaces of the structure, ii) distance to the origin of the fault plane from an arbitrarily chosen reference on the profile, and iii) the geometry of the fault plane. Based on the proposed methodology, a GUI based software, GRIN2DFL, coded in JAVA to analyze the gravity anomalies is developed. Finally, the applicability of the proposed inversion is demonstrated on both synthetic and real field gravity anomalies.

2.3 Forward gravity modeling – Theoretical considerations

In a Cartesian coordinate system the geometry of listric fault model, shown in xz cross-section in Figure 2.1, is bounded by upper and lower horizontal planes z = z_T, and z_B, semi-infinite in the positive x-axis direction, bounded on the other side by a non-planar fault ramp defined by a function, $\zeta(z)$, and extends to infinity in the positive and negative y-axis directions. Presuming that the footwall remains undeformed and intact, the gravity anomaly of such a structure, $g_{mod}(X_j, Z_j)$, at any point, $P(X_j, Z_j)$, on a profile,

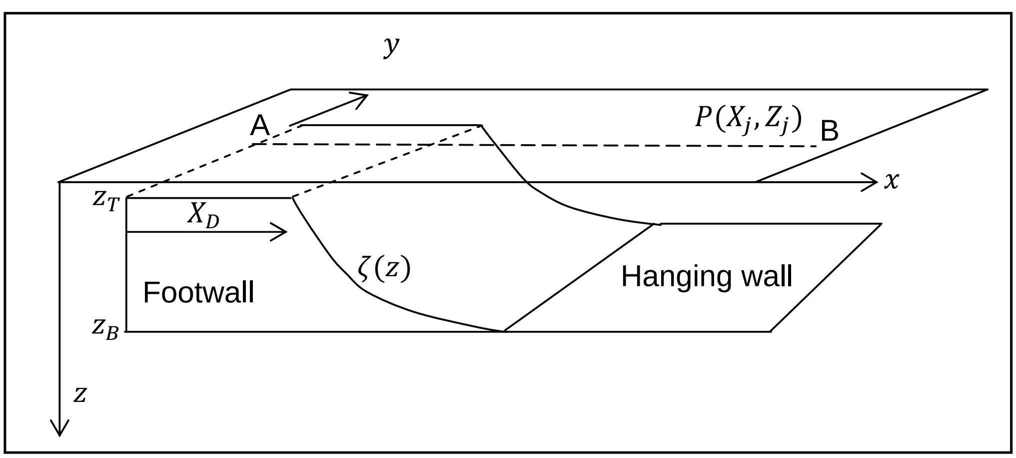


Figure 2.1 Schematic representation of a 2D listric fault source. AB is a profile across the strike along which the interpretation is intended

AB, outside the source region can be expressed as (Chakravarthi et al., 2015a),

$$g_{mod}(X_j, Z_j) = 2G \int_S \frac{\Delta\rho(z)\overline{z - Z_J}dxdz}{\overline{x - X_J}^2 + \overline{z - Z_J}^2}, \tag{2.1}$$

where, $dxdz$ is the cross-sectional area of a representative element within the source; whose coordinates are given by (x, z). Also, $\Delta\rho(z)$ is the density contrast of sediments at any depth, z, within the hanging wall system represented by equation (1.14).

Substituting equation (1.14) for $\Delta\rho(z)$ and upon integration with respect to x, equation (2.1) becomes

$$g_{mod}(X_j, Z_j) = 2G\Delta\rho_0 \int_{z_T}^{z_B} e^{-\lambda z} \left\{\frac{\pi}{2} - tan^{-1} \langle\frac{\zeta(z) - X_j}{z - Z_j}\rangle\right\} dz, \tag{2.2}$$

where,

$$\zeta(z) = \sum_{k=0}^{N1} f_k z^k. \tag{2.3}$$

Here, f_k represents a set of coefficients and $N1$ stands for the degree of polynomial. It is to be noted that closed form analytical solution does not exist for equation (2.2) in the space domain; however, it can be solvable by means of a numerical integration. In this case, the Simpson's rule is used to solve equation (2.2). This method of solving the equation has an added advantage that it could overcome possible singularities. It is obvious that by letting λ to zero in equation (2.2), gravity anomalies of the structure with uniform density can be realized.

2.4 Inversion of gravity anomalies

Inversion of gravity anomalies of listric fault sources is tantamount to a mathematical exercise of trying to fit the modeled gravity anomalies, $g_{mod}(X_k, Z_k)$, $k = 1, 2, .., N_{obs}$, to the observed ones, $g_{obs}(X_k, Z_k)$, $k = 1, 2 \ldots, N_{obs}$, in an iterative approach employing the principles of least squares. Here, N_{obs} stands for the number of observations. For the optimum parameters the modeled anomalies closely mimic the observed ones.

The advantage of the present algorithm is that it estimates the initial parameters of the source based on a few selective anomalies and subsequently improves them automatically by minimizing the errors between the observed and model gravity anomalies in an iterative approach.

To start with, the algorithm estimates one half of the maximum anomalous field and finds the corresponding distance, X_D, (Figure 2.1) on the profile from a chosen reference (Murthy, 1998; Chakravarthi et al., 2015a). The algorithm assigns this parameter value to the constant term, f_0, of the polynomial, $\zeta(Z)$, and sets the remaining coefficients to zero. Approximate depth to the bottom of the structure, z_B, is calculated based on the Bouguer slab formula (Cordell, 1973; Chakravarthi et al., 2015a) as

$$z_B = \frac{-1}{\lambda} log\left[1 - \frac{\lambda g_{obsmx}}{2\pi G \Delta\rho_0}\right], \qquad (2.4)$$

where, g_{obsmx} is the maximum observed gravity anomaly (absolute) on the profile. A small value of 1E-04 km is assigned to the depth parameter, z_T.

Using these initial parameters of the structure the algorithm calculates the modeled gravity anomalies, $g_{mod}(X_j, Z_j),\ j = 1, 2, \dots, N_{obs}$ using equation (2.2) and quantifies the misfit (J) between the observed and modeled anomalies based on the expression

$$J = \sqrt{\frac{\sum_{j=1}^{N_{obs}}\left[ERR(x_{j,}z_j)\right]^2}{N_{obs}}}, \tag{2.5}$$

where, $Err(X_j, Z_j) = g_{obs}(X_j, Z_j) - g_{mod}(X_j, Z_j),\ j = 1, 2, \dots\dots, N_{obs}$.

The difference between the observed and modeled gravity anomalies at any observation, (X_j, Z_j), on the profile can be expressed as a cumulative of the products of partial derivative of the anomaly with respect to each unknown parameter to be estimated and corresponding increments as (Chakravarthi et al., 2015a)

$$g_{obs}(X_j, Z_j) - g_{mod}(X_j, Z_j)$$
$$= \frac{\partial g_{mod}(X_j, Z_j)}{\partial z_B} dz_B + \sum_{k=0}^{N1} \frac{\partial g_{mod}(X_j, Z_j)}{\partial f_k} df_k, \tag{2.6}$$

where, dz_B is an increment/decrement in z_B and $df_k,\ k = 0, 1, 2, .., N1$ are increments/decrements in f_k respectively. Similar linear equations are constructed for all observations, $(X_j, Z_j),\ j = 1, 2, .., N_{obs}$ on the profile and $(N1 + 2)$ normal equations are framed and solved by minimizing the misfit given in equation (2.5) using the ridge regression algorithm (Marquardt, 1970). The relevant system of normal equations is expressed in a matrix form as (Chakravarthi et al., 2015a)

$$(A + \delta I)T = S, \qquad (2.7)$$

where, A is nxn matrix whose elements $A_{mj'}$ are given by

$$A_{mj'} = \sum_{j}^{N_{obs}} \sum_{m=1}^{N1+2} \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_{j'}} \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_m}, j' = 1, 2, \ldots, (N1+2) \qquad (2.8)$$

$$B = \sum_{j=1}^{N_{obs}} Err(X_j, Z_j) \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_{j'}}, j' = 1, 2, \ldots, (N1+2). \qquad (2.9)$$

$$T = da_m, m = 1, 2, \ldots N1 + 2 \qquad (2.10)$$

Further, $a_1 = z_{B,} \quad a_m = f_{m-2}$ for $m = 2\ to\ N1 + 2$. Also, δ is a damping factor and I is a diagonal matrix containing the diagonal elements of the matrix, A. The algorithm computes the partial derivatives required in equation (2.8) and (2.9) by a numerical method (Chakravarthi et al., 2001; Chakravarthi et al., 2015a).

The inversion algorithm initially assigns a value of 0.5 to the damping factor, δ, and solves equation (2.7) for the increments/decrements, da_m, $m = 1\ to\ N1 + 2$. These values are added to/subtracted from the existing parameters, a_m, to obtain the improved parameters, $a_m^{'}$, for $m = 1, 2, .., N1 + 2$. If the resulting misfit, J_{mod}, (equation 2.5) obtained with the improved parameters is less than its previous value, J, then the algorithm assigns J_{mod} to J and $a_m^{'}$ to a_m and the present value of the damping factor, δ, is further decreased by a factor of 1/2. If J_{mod} is greater than J at any stage during the process of inversion then the current value of δ is doubled and equation (2.7) is again solved for the parameters, . These values are added to/subtracted from the existing parameters, , till attains a value less than or equal to . The algorithm repeats the process till the specified number of iterations is completed or the misfit (equation 2.5) becomes less than the predefined allowable error or the damping factor, , assumes an unusually large value (Chakravarthi, 2003; Chakravarthi et al., 2015a). The parameters , and remain stable during the process of inversion.

2.5 Description of the software – GRIN2DFL

A GUI based software, GRIN2DFL, coded in JAVA is developed based on the algorithm described in section 2.4 to analyze the gravity anomalies produced by 2D listric fault sources using EDCM (Annexure 2-A). The software is built on the Model-View-Controller (MVC) architecture according to the structural relationship shown in Figure 2.2.

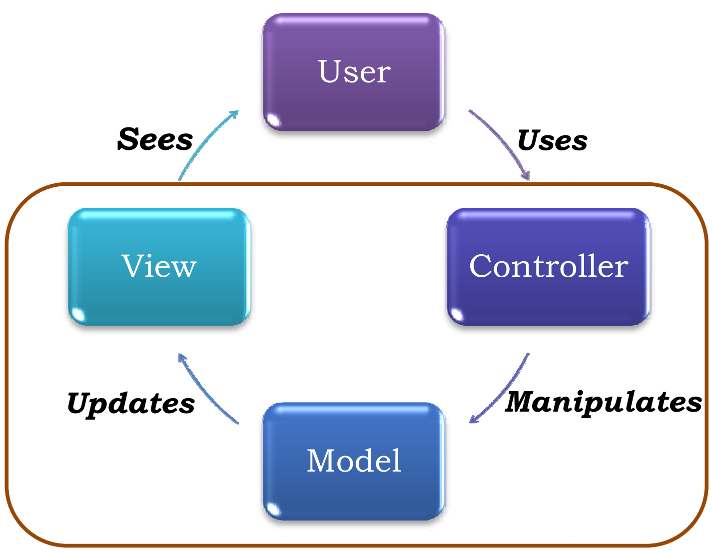


Figure 2.2 Structural relationships between Model, View and Controller

The module ‘Model’ (Figure 2.2) performs the task of finding the approximate location of the fault plane, computes the gravity anomalies of the structure and performs the business logic of the inversion algorithm. The ‘View’ module reads the input parameters as specified by the user and shows the interpreted results as output subsequent to inversion. The role of ‘Controller’ is to pass on the required actions to the view and model modules.

Upon invoking the batch file of the software, the view module appears on the monitor as shown in Figure 2.3.

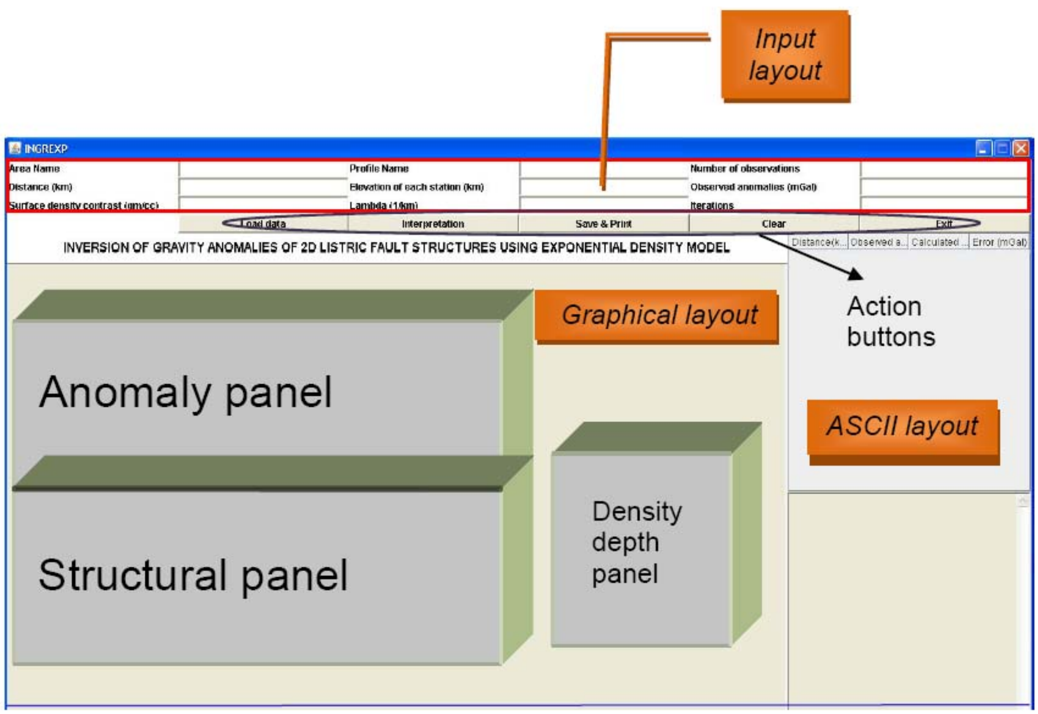


Figure 2.3 View module of INGREXP

The view module is structured into three layouts, namely i) input, ii) graphical, and iii) ASCII as shown in Figure 2.3. The input layout consists of nine fields and five action buttons. The graphical layout, which serves as an interface between the user and the software, is further divided into the anomaly panel on top, structure panel in bottom, misfit and density-depth panels in the right. The ASCII layout displays the interpreted results in ASCII form. The user enters the information pertaining to the area name, profile name, number of observations, distance to each observation (km), station elevation (km), observed gravity anomalies (mGal), constants of EDCM namely $\Delta\rho_0$ (gm/cm^3) and λ (km^{-1}), and number of iterations to be performed in respective fields in the input layout (Figure 2.3) and opts for the analysis by using the action button - Interpretation. Alternatively, the user can enter and save the input data in a Microsoft Office Excel sheet and invokes the same to the software by means of 'Load data' action button (Figure 2.3). To avail such an option of inputting the data in an Excel sheet and to load it to the software, the user needs to download the executable Jar File (http://www.java2s.com/Code/Jar/j/ Downloadjxl26jar.htm-jxl-2.6.jar) in the root directory followed by setting up a class path in the batch file. 'Save and Print' action button enables the user to save the interpreted results and allows for printing.

The advantage and key features of the software are i) it is fully automatic, ii) it works on any operating system (platform independent) with at least jdk 1.6 version installed, and iii) it facilitates the user to visualize the improvements in the modeled space and corresponding model gravity anomalies, changes in the misfit, and variation of density contrast with depth in animated forms during the process of inversion.

2.6 Applications

Reliability and applicability of the inversion technique and the software are demonstrated with two examples, one synthetic and a real. The estimated parameters subsequent to inversion are compared with the assumed (actual) parameters in case of synthetic example and with available information in case of the field example. In both cases, the observer is on top of the topography at Z_j =0.

2.6.1 Synthetic example-Inversion of noisy gravity anomalies

Noisy gravity anomalies produced by a synthetic model of a listric fault source (Figure 2.4b) at 21equispaced observations in the interval, $X_j \in$ [20 km, 40 km], is shown in Figure 2.4a. In this case the fault ramp is exposed to the surface at 30th km on the profile (Figure 2.4b). The parameters assumed to generate the gravity anomalies of the source are: z_B = 2.0 km, $\Delta\rho_0$ = -0.35 gm/cm^3, λ = 0.4 km^{-1}. The prescribed density contrast variation with depth is shown in the inset of Figure 2.4b. Further, a 7th degree polynomial with a set of 8 arbitrarily chosen coefficients (Table 2.1) simulates the geometry of the fault plane with depth (Figure 2.4b). In this case, pseudorandom noise present in the anomaly is Gaussian with zero mean and a standard deviation of 0.27 mGal. Treating the noisy data (shown as a solid line Figure 2.4a) as the observed anomalies, the inversion was performed by the present technique to recover the fault structure.

2D Inversion-Synthetic example

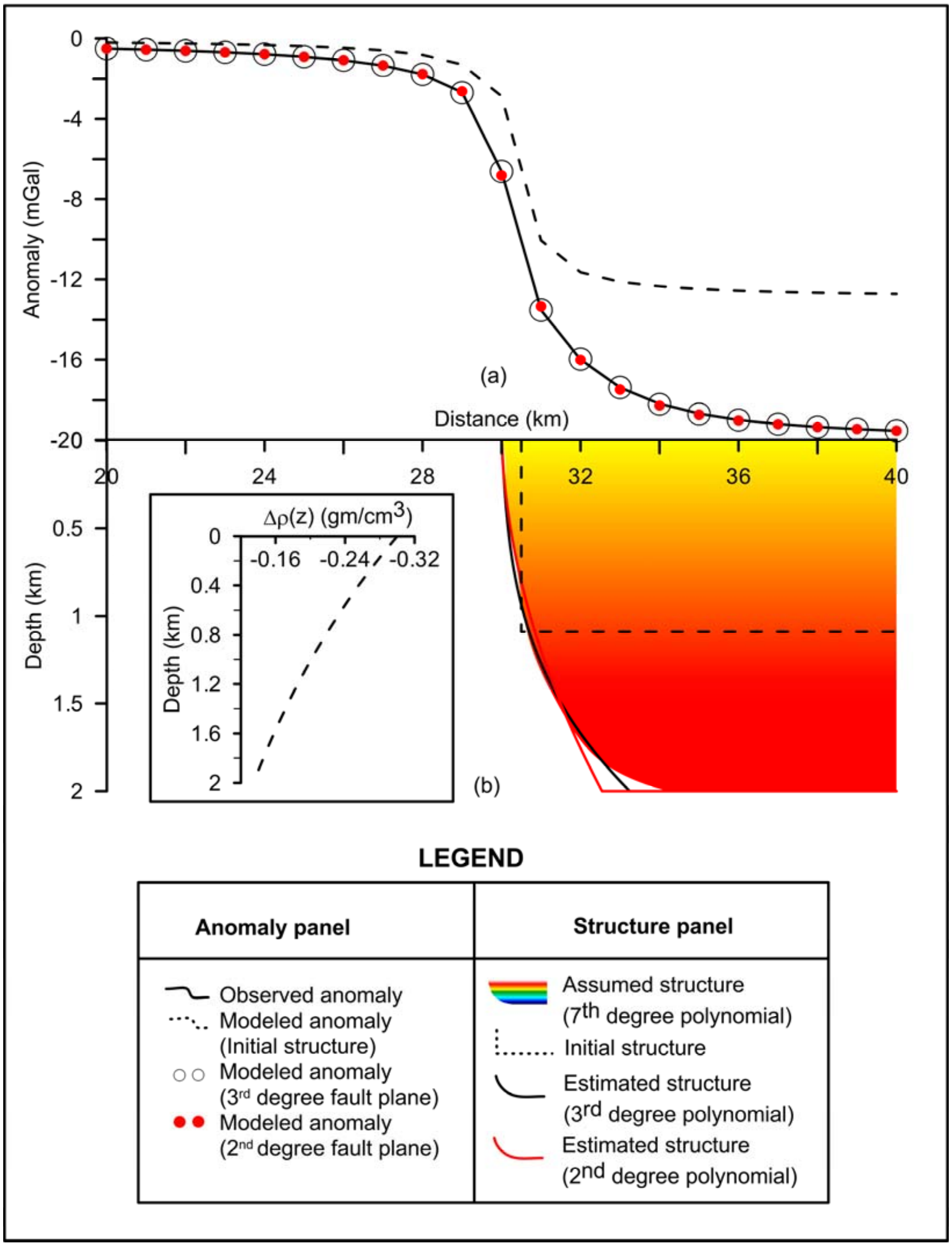


Figure 2.4 (a) Observed and modeled gravity anomalies, (b) assumed, initial and modeled structures. Prescribed Exponential Density Contrast Model (EDCM) is shown in the inset of (b). Colour gradation from yellow to red within the hanging wall represents increase in density

Having consider the fact that the degree of polynomial to be chosen to describe the fault plane geometry is always prone to be uncertain in the absence of additional source of information; two polynomials namely a 2nd degree and a 3rd degree are considered in the inversion to recover the fault plane geometry (it is to be realized that a 7th degree polynomial is used to describe the fault plane while generating the noisy anomalies). For such an approximation, the algorithm had computed the initial thickness of the structure as 1.09 km and the initial parameter value of the coefficient, f_0, of the polynomial, $\zeta(z)$, as 30.5 in each case (Figure 2.4 b).

Table 2.1
Assumed and estimated coefficients of polynomials, synthetic example.

Coefficient	Assumed (7th degree polynomial)	Estimated (3rd degree polynomial)	Estimated (2nd degree polynomial)
f_0	29.994	30.0024	29.9999
f_1	1.963	0.2592	0.1842
f_2	-13.978	-0.0518	0.5440
f_3	45.868	0.3666	
f_4	-70.937		
f_5	56.408		
f_6	-22.189		
f_7	3.432		

For a 2nd degree polynomial approximation of the fault plane, the algorithm had performed 16 iterations and for a 3rd degree 41 iterations respectively. Further, in case of 2nd degree polynomial the damping factor, δ, after the 16th iteration has attained a larger value than its preceding one thereby the algorithm got terminated. On the other hand, for a 3rd degree polynomial approximation, the misfit fell below a predefined allowable error of 0.003 at the end of the 41st iteration. By and large, in either case the nature of fit between the observed and modeled gravity anomalies is satisfactory (Figure 2.4a). The error between observed and modeled gravity anomalies at the end of respective concluding iterations in both cases are shown in Figure 2.5a. The parameter values of the estimated coefficients of the two polynomials at the end of respective concluding iterations were given in Table 2.1. Figure 2.4b shows the geometries of fault structures recovered in either case after respective inversions. Error analysis of misfit, coefficients of selected polynomials and the thickness of the fault morphology versus the iteration number are shown in Figure 2.5b.

The predicted depth to the floor of the structure (decollement horizon) from present inversion in either case was 1.99 km, which agrees well with the assumed depth of 2 km (Figure 2.4b). A negligible error of 0.5% between the assumed and predicted depths is acceptable in both cases because the anomalies used in inversion are noisy.

Based on the analysis, it is concluded that the estimated thickness of a listric fault morphology is independent on the choice of the degree of polynomial used in the inversion to describe the fault plane geometry. However, the use of a lower order polynomial in the inversion would result in the underestimation of the amount of extension across the fault ramp (Figure 2.4b).

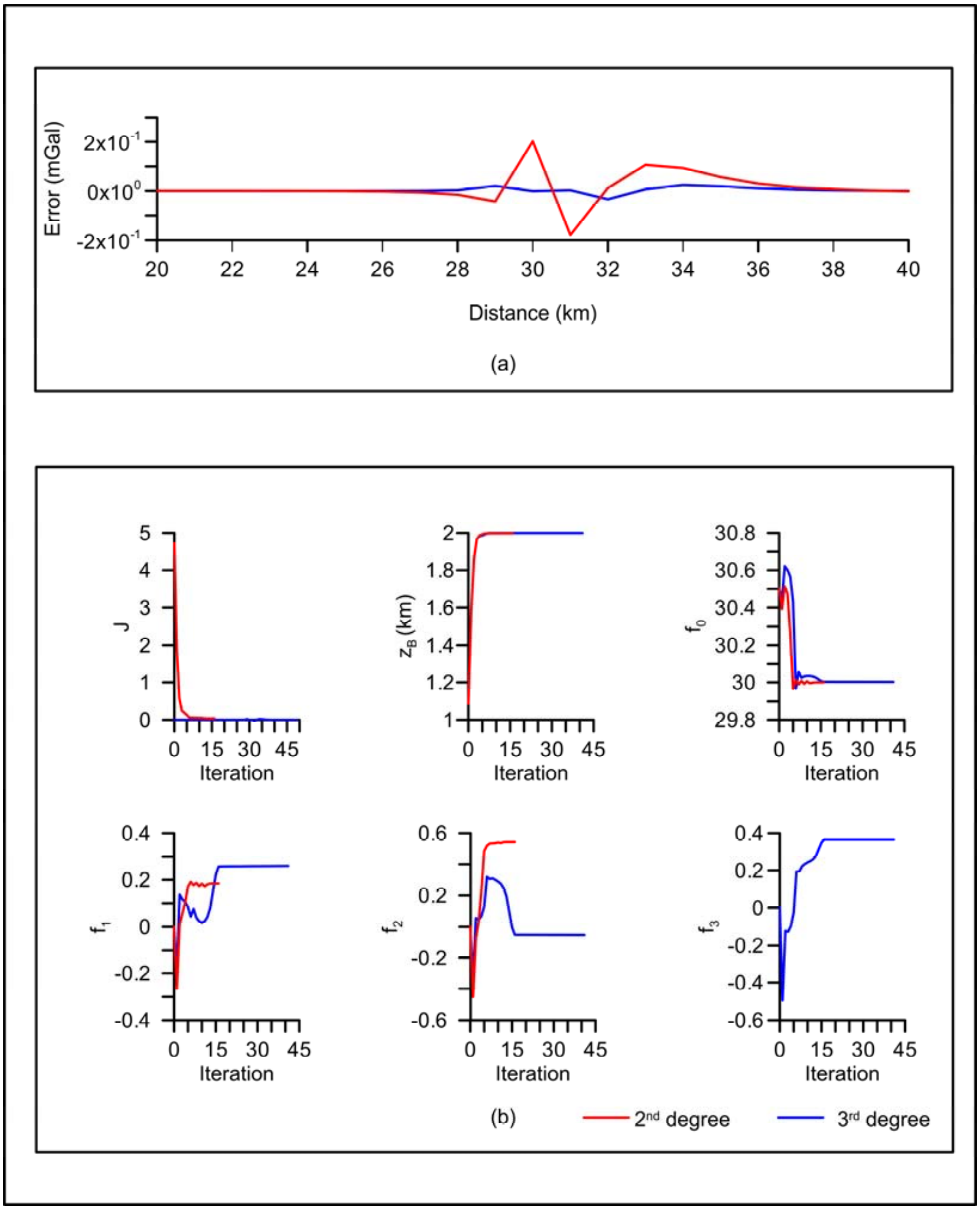


Figure 2.5 (a) Error analysis of the gravity anomaly, (b) variation of misfit and various shape parameters with iteration number

2.6.2 Field example

Narmada-Son-Damodar (NSD), Pranhita-Godavari (PG) and Mahanadi (M) river valleys are the main repositories in the peninsular India, where Gondwana succession has been preserved. Based on the geology, structure, and nature of lithic fill, Murthy and Parthasarathy (1988) have distinguished the Pranhita-Godavari Valley into four sub-basins namely; Godavari, Kothagudem, Chintalpudi, and Krishna-Godavari respectively. Out of the four, the Godavari sub-basin covers an area of about 12,350 sq km over a strike length of approximately 200 km (Figure 2.6). The sub-basin exhibits a half-graben structure defined by the Ahiri-Cherla master fault on the northeastern side (Qureshy et al., 1968; Ramakrishna and Chayanulu, 1988).

Mishra et al. (1987) had discussed in detail the gravity survey of the Pranhita-Godavari Valley including the distribution of gravity observations, application of various corrections to the measured data and the accuracy of the Bouguer gravity anomalies. The steep gradient observed on the Bouguer anomaly on the eastern side of the basin throughout its strike is associated with the Ahri-Cherla master fault (Figure 2.6). Chakravarthi and Sundararajan (2004) have analyzed the gravity anomalies of this master fault along a selected profile treating the fault plane of the structure as a planar surface. In the present case we analyze the gravity anomalies of the master fault along a profile, AA', (Location of the profile is shown in Figure 2.6) presuming the fault plane of the structure as non-planar. The smoothed residual gravity anomaly across the fault structure along the profile, AA', obtained after separating a regional trend is

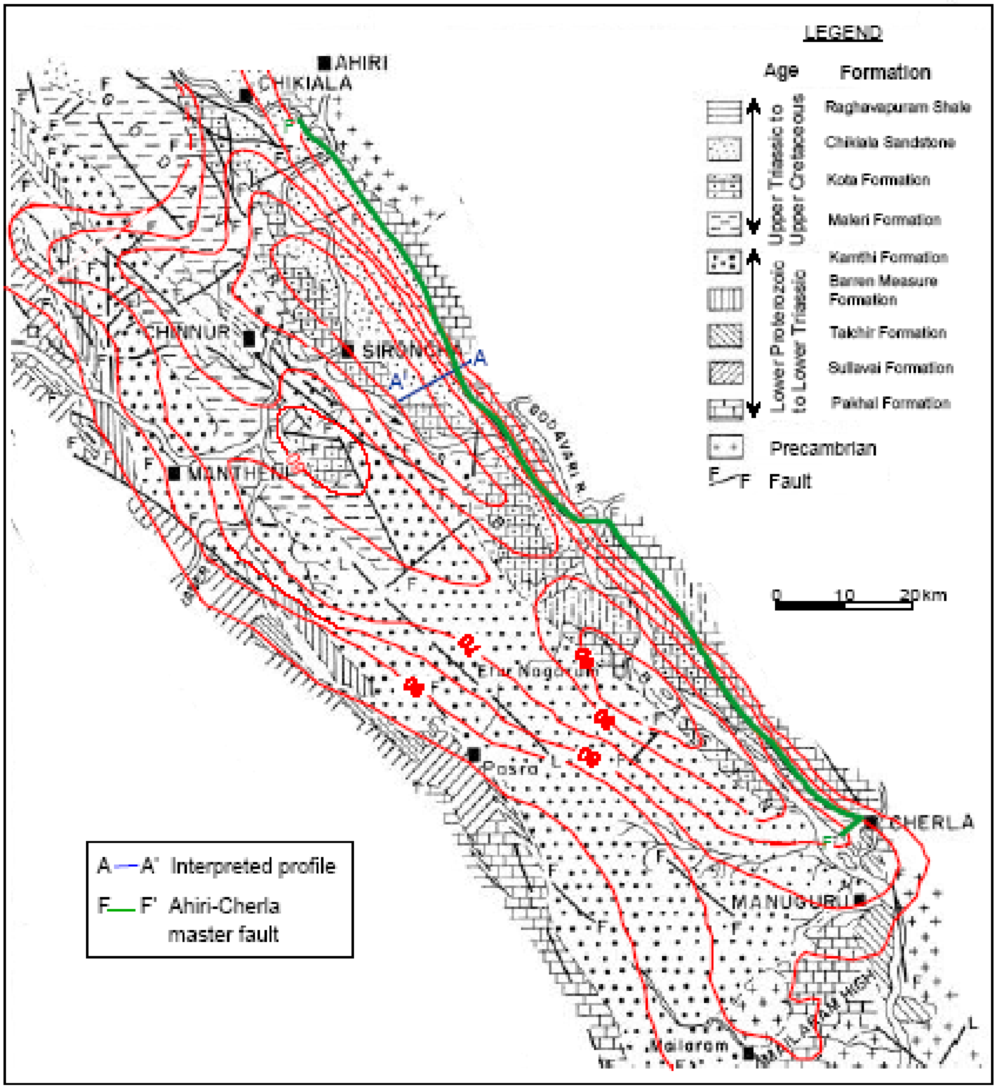


Figure 2.6 Geologic and Bouguer gravity anomaly map of the Godavari sub-basin, India (after Chakravarthi and Sundararajan, 2004). Note that anomaly values have to be read with a negative sign

shown as a solid line in black in Figure 2.7a. The exponential density contrast model (EDCM) obtained by fitting equation (1.14) to the derived parabolic density model of the basin (Chakravarthi and Sundararajan, 2004) was used to analyze the anomalies shown in Figure 2.7a. The fitted EDCM (inset of Figure 2.7b) is given by (Chakravarthi et al., 2015a)

$$\Delta\rho(z) = -0.4554e^{-0.3929Z}. \tag{2.11}$$

As in the case of synthetic example, the inversion was performed on the observed anomalies of the sub-basin considering both 2nd and 3rd degree polynomial approximations for the fault plane. The initial parameter value of z_B in each case is 1.44 km. The dashed line shown in Figure 2.7b represents the starting model of fault geometry based on the initial parameters estimated by the algorithm for f_0 (Table 2.2) and z_B.

Table 2.2

Estimated coefficients of polynomials, Ahiri-Cherla master fault Godavari sub-basin, India.

Coefficient	**Estimated (3rd degree polynomial)**	**Estimated (2nd degree polynomial)**
f_0	4.9698	4.9693
f_1	0.2226	0.2259
f_2	0.0872	0.0842
f_3	-0.0006	

The algorithm took 21 iterations to achieve proper convergence in case of a 2nd degree polynomial approximation for the fault plane and 47 iterations in case of 3rd degree approximation respectively. The modeled gravity anomaly

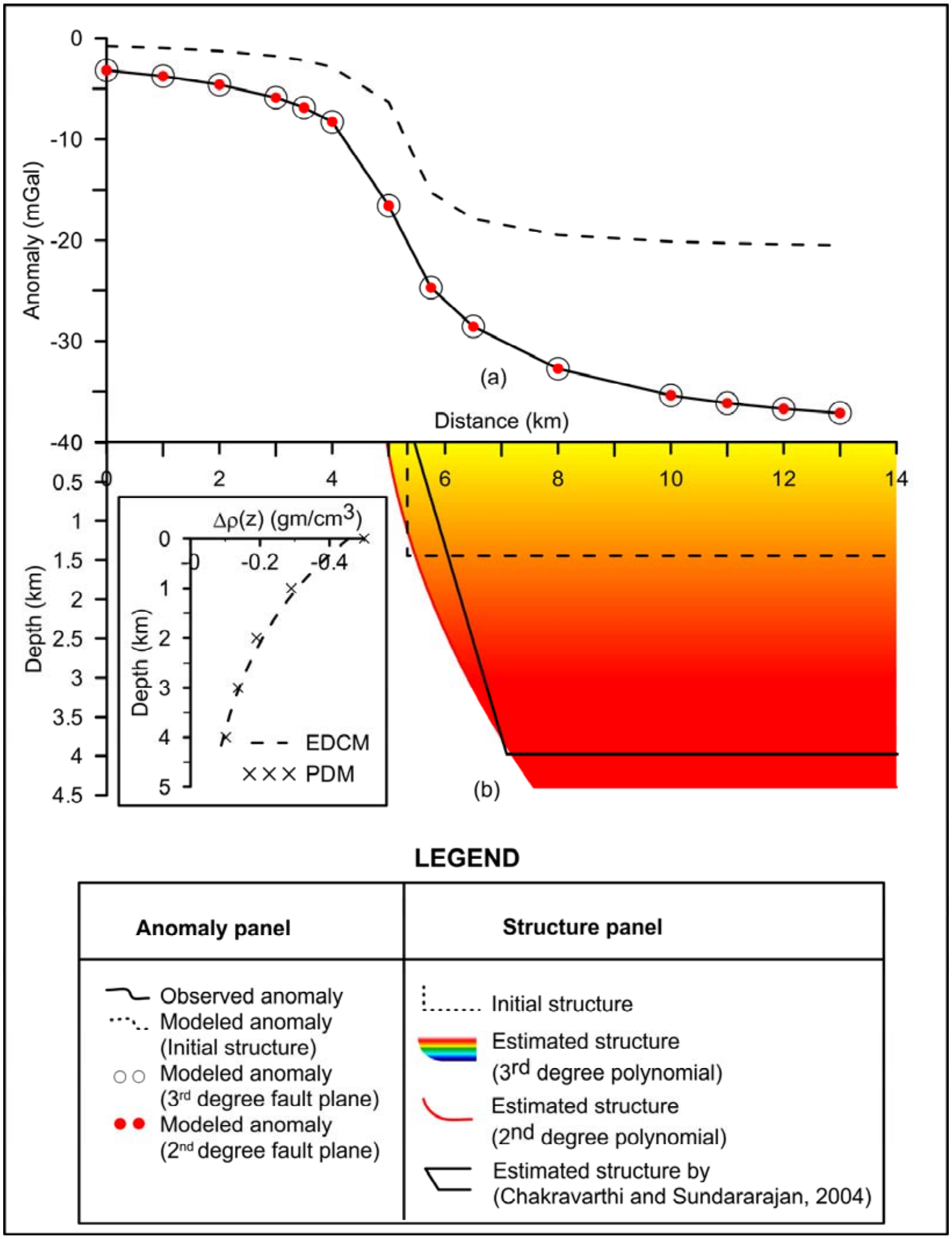


Figure 2.7 (a) Observed, initial and modeled gravity anomalies, (b) initial and estimated structures, Ahiri-Cherla master fault, Godavari sub-basin, India. Simulated exponential density contrast model is shown in the inset of (b)

subsequent to respective concluding iterations closely mimic the observed one (Figure 2.7a). The magnitude of residuals between the observed and modeled anomalies in either case is below $\mp$ 0.095 mGals (Figure 2.8a). The estimated parameter values of the coefficients for 2nd and 3rd degree polynomials after respective inversions are given in Table 2.2.

The estimated depth to the bottom of the fault structure in either case is 4.39 km, which again confirms the fact that the choice of the degree of polynomial in the inversion to describe the fault plane geometry does not appreciably affect the interpreted depth. Furthermore, the inferred structure of the Ahiri-Cherla fault morphology based on a 3rd degree polynomial approximation of the fault plane almost coincides the structure estimated using a 2nd degree polynomial (Figure 2.7b). The changes in misfit and other estimated parameters against the iteration are shown in Figure 2.8b in each case. The fact that the inferred thickness of the fault morphology from present inversion (4.39 km) closely matches with an estimated fault throw of 4.5 km by Murthy and Parthasarathy (1988) and about 4 km by Chakravarthi and Sundararajan (2004) demonstrates the practical applicability of the technique.

The interpreted geometry of the fault structure along the same profile with a planar fault surface approximation (Chakravarthi and Sundararajan, 2004) is also shown in Figure 2.7b (solid line) for comparison. It is noticed from Figure 2.7b that the interpreted dips of the fault plane from both the present method and the one reported by Chakravarthi and Sundararajan (2004) are correlated well with each other up to a depth of 1.5 km, beyond which the

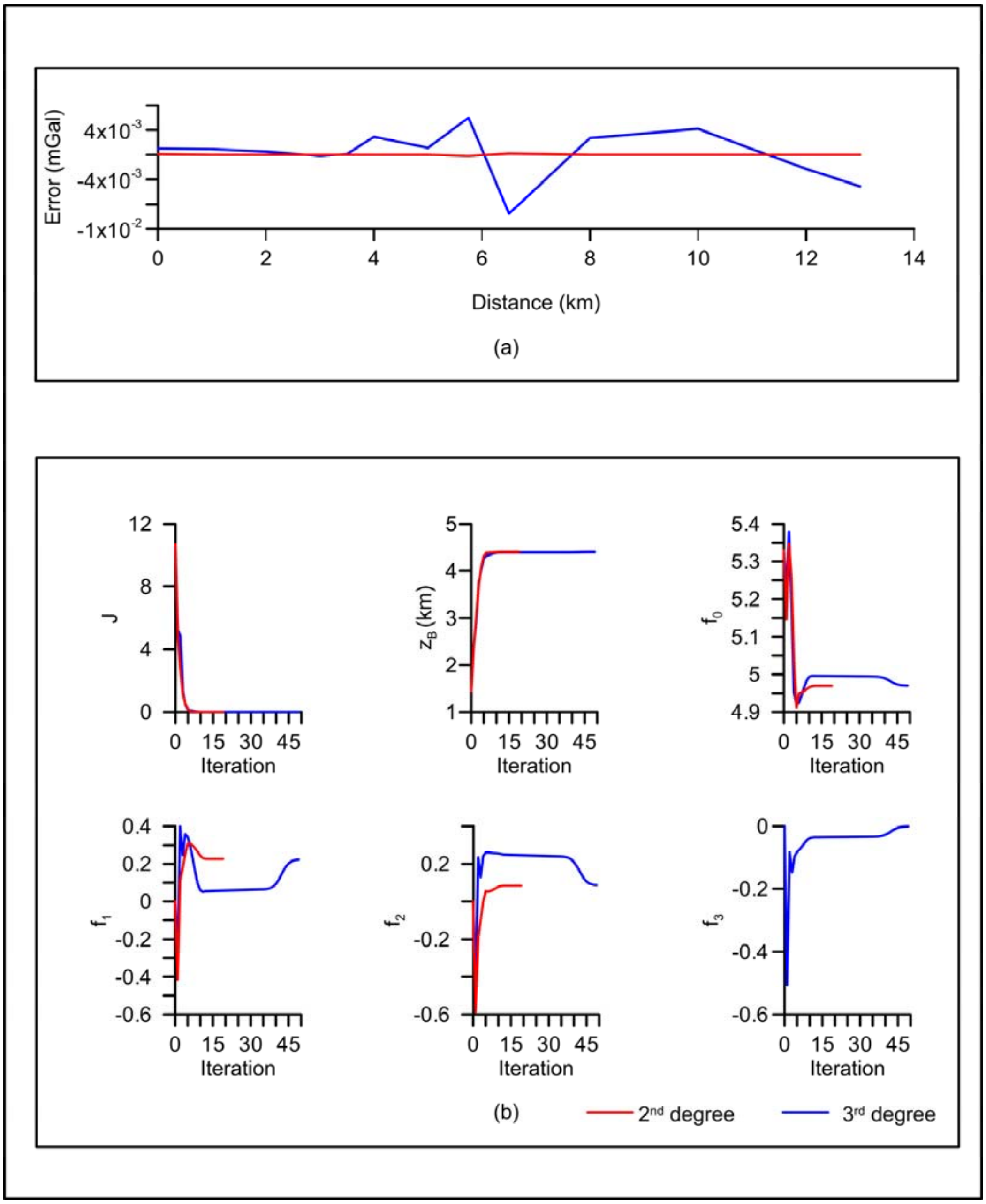


Figure 2.8 (a) Error analysis of the gravity anomaly, (b) variation of misfit and various shape parameters with iteration number, Ahiri-Cherla master fault, Godavari sub-basin, India.

estimated fault plane from the present method portrays progressively decreasing dips.

2.7 Results and discussion

Although listric fault morphologies associated with thick sedimentary sequence of rocks warrants the use of exponential density contrast model in the analysis of gravity anomalies to obtain more reliable results, not many methods have been reported because of the fact that no closed form gravity expressions could be derivable in the space domain to realize forward modeling. A new method combining both analytical and numerical approaches has been developed in the space domain to realize forward gravity modeling of 2D listric fault morphologies using an exponential density contrast model. Prescribed polynomial functions of arbitrary but specific degree are used to describe the non-planar fault ramps. An automatic inversion technique coupled with relevant GUI based software is developed to analyze the gravity anomalies for the unknown parameters of fault morphology. The application of the technique is exemplified with both synthetic and real field gravity anomalies.

In case of a synthetic fault model, a 7^{th} degree polynomial is used to describe the fault ramp to compute the gravity response. Pseudorandom noise with zero mean and standard deviation of 0.27 mGal is added to the residual gravity anomaly produced by the structure before being subjected for inversion by the proposed technique. Also, as uncertainty prevails on the selection of the degree of polynomial to be chosen to simulate the fault plane in the inversion (in the absence of additional sub-surface information), a 2nd and a 3rd polynomials are assumed, independently, to study their affect on the interpretation, if any. It was found from the analysis that the estimated depth of the structure remains more or less the same irrespective of the degree of polynomial used in the inversion (whether a 2nd degree or a 3rd degree) to describe the fault ramp. However, it is shown that the choice of lower order polynomials in the inversion would lead to underestimate the amount of extension across the fault ramp.

The analysis of the observed gravity anomalies across the Ahri-Cherla master fault (boundary fault) of the Godavari sub-basin in India using the derived exponential density contrast model has yielded an interpretation that is consistent with the available/reported information.

In short, the successful application of the proposed technique on both synthetic and real world gravity anomalies testifies its applicability.

CHAPTER THREE

Interactive gravity modeling of 2.5D strike listric fault sources with arbitrarily varying density-depth relationship*

3.1 General:

Modeling of concealed density interfaces from observed gravity anomalies is one of the classic geologic applications of the gravity method. Chakraborty et al. (2003), Chakravarthi (2011b) have shown that the marginal/ boundary faults associated with some of the sedimentary basins, such as the pull-apart basins, are not only listric in nature but also posses finite strike lengths (see for e.g., Hutar, Auranga, Karanpura, Bokaro, Jharia, and Ranigunj basins of Damodar valley in India). Peirce and Lipkov (1988) have also showed that the faults on the southwestern margin of the Rukwa Rift have finite strike lengths. Therefore, approximating such listric fault morphologies to appropriate geophysical models having finite strike lengths is justified to analyze the gravity anomalies. However, 3D models are more expensive in terms of data requirement and computational time than 2D; therefore, the use

---

** Published in Journal of the Geological Society of India (2014, 83, 577-585), Springer.*

of 2.5 dimensionality (2.5D) in the interpretation of gravity anomalies is more appropriate.

3.2 Status of existing interpretational techniques

Rao and Prakash (1990) have considered 2.5 dimensionality for the fault structures and developed an interpretation algorithm to analyze the gravity anomalies arise from such models using a quadratic density-depth relationship. This technique, being efficient on residual gravity anomalies, may yield unreliable interpretations in the presence of regional gravity background. In addition, this density function would pose problems in automatic modeling of gravity anomalies as elucidated by Chakravarthi and Sundararajan (2006). On the other hand, Chakravarthi (2008) had used a parabolic density model to devise an algorithm to estimate simultaneously the parameters of 2.5D strike fault structures and regional gravity background from a set of observed Bouguer gravity anomalies. Nevertheless, the practical applicability of the enlisted methods is limited because these techniques presume planar surfaces for fault planes, which in reality may not be so as large dip-slip faults are usually curved in cross-section (Peirce and Lipkov, 1988; Brun and Choukroune, 1983; Chakravarthi, 2011b; Chakravarthi et al., 2014).

Martín-Atienza and García-Abdeslem (1999) have also developed a technique based a quadratic density model to compute the gravity anomalies of geologic sources bounded either laterally or vertically by continuous functions. Even though, this method can be used to simulate the geometries of listric fault sources to compute the gravity anomalies, again it is efficient only for 2D sources. In recent past, Chakravarthi (2010a) devised a method to compute the gravity responses of 2.5D listric fault sources; where the fault planes are described by non-planar surfaces, detached hanging wall system composed of thick-sectioned sediments and the density contrast of which varies continuously with depth following a parabolic equation. Later, Chakravarthi (2011b) developed an automatic inversion to estimate the geometries of 2.5D strike listric fault sources and regional gravity background from the observed Bouguer gravity anomalies.

The existing density functions including those enlisted above will fail to describe the sub-surface density distribution in case the detached hanging wall consists in both high and low-density formations. Chakravarthi (2010b) devised a strategy coupled with a code to realize forward gravity modeling of such listric fault sources. In order to calculate the gravity anomalies using this method, the coefficients of the polynomial to describe a fault plane need to be estimated separately and supplied to the code as a part of input along with the depths and densities of various sub-surface formations. The code then generates the output in ASCII format. In reality, one needs to improve the model parameters several times until the modeled gravity anomalies mimic the observed ones. It becomes a prodigious task to input several such combinations of model parameters to realize gravity modeling.

Therefore, a need exists to develop a new interpretation strategy with relevant software that could overcome the drawbacks associated with the

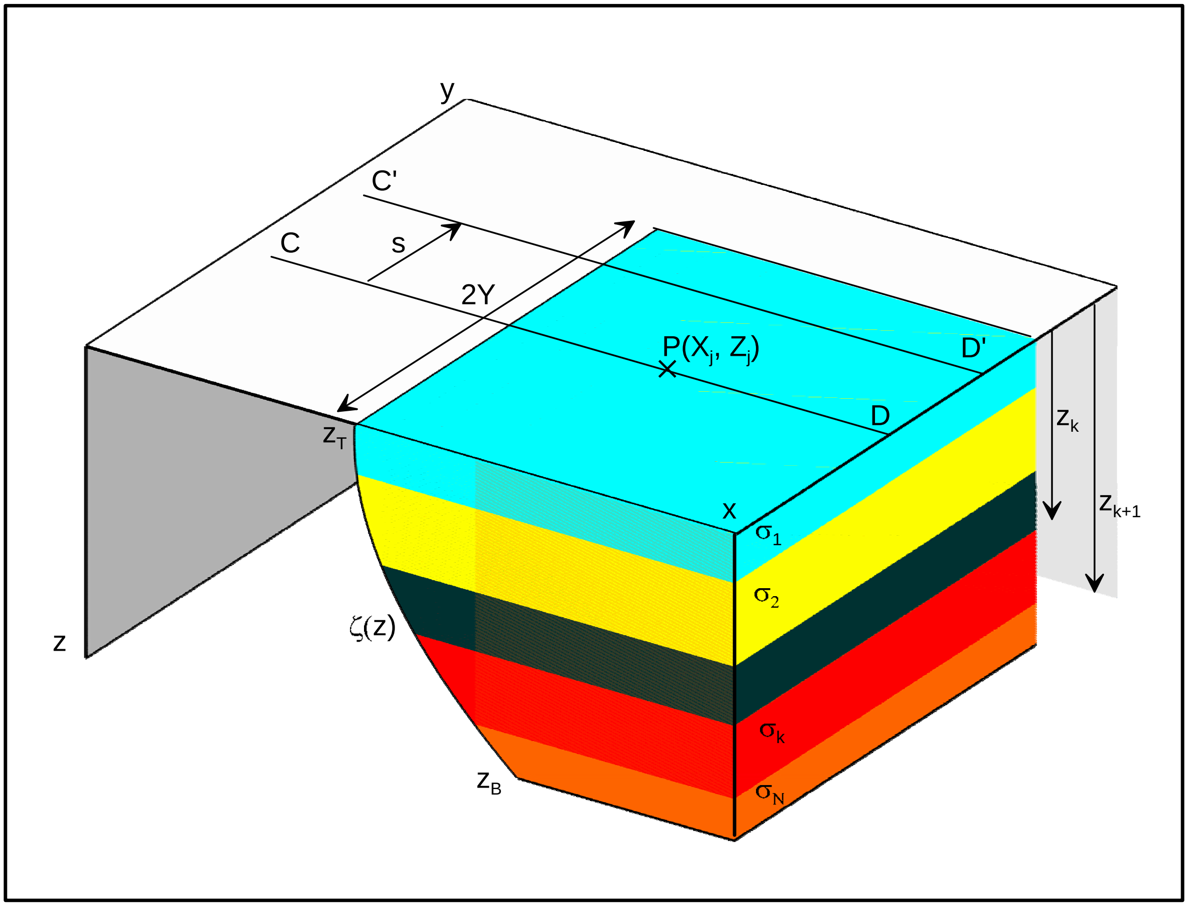


Figure 3.1 Schematic representation of a strike limited listric fault source. The detached downthrown block (hanging wall) is consisting of N number formations with differing thicknesses and densities. The limited strike length prevents the structure to represent as a 2D source.

existing algorithms to analyze the gravity anomalies of strike limited listric fault morphologies, where the detached hanging wall contains in both high and low density formations. In addition, the interpretation tool should be simple and user-friendly in terms of interactive model construction and subsequent modification to realize modeling in real time.

3.3 Forward gravity modeling – Theoretical considerations

In Cartesian co-ordinate system, let the z-axis be positive vertically downwards and the x-axis transverse to the strike of a listric fault source, whose geometry is shown in Figure 3.1. The structure is confined between the limits, z_T and z_B, along the z-axis, and along the x-axis it is bounded by a function, $\zeta(z)$, on the left and towards the right it is extending to infinity. Further, the structure is having a limited strike length of $2Y$ along the y-axis transverse to the xz plane. The detached hanging wall of the structure is presumed to consist in several geologic formations, N in number; where each one has its own density, σ_i, and thickness, z_i, $i = 1, 2, ., N$. For such a structure, the gravity anomaly, $g_{mod}(X_j, Z_j)$, at any point, $P(X_j, Z_j)$, on a profile, CD, that runs along the x-axis and bisects the strike length, $2Y$, of the structure outside the source region (Figure 3.1) is given by Chakravarthi et al., (2014) as

$$g_{mod}(X_j, Z_j) = 2G \sum_{k=1}^{N} \Delta\rho_k \int_{z_k}^{z_{k+1}} \left[tan^{-1} \langle \frac{Y}{z - Z_j} \rangle \right.$$

$$\left. - \, tan^{-1} \langle \frac{Y \, \overline{\zeta(z) - X_J}}{\overline{z - Z_J} \sqrt{\overline{\zeta(z) - X_J}^2 + \overline{z - Z_J}^2 + Y^2}} \rangle \right] dz, \qquad (3.1)$$

where, z_k and z_{k+1} are depths to the top and bottom of the k^{th} formation within the hanging wall, and $\zeta(z)$ is represented by the equation (2.3). It is convenient to solve equation (3.1) numerically rather than analytically because the polynomial, $\zeta(z)$, may take any degree (Chakravarthi et al., 2014). In case the profile does not bisect the strike length of the structure but runs at an offset, s, (such as the profile, C'D' in Figure 3.1) then the anomalous field at any point on the profile can be calculated as the average of equation (3.1) by substituting, Y-s and Y+s for Y. Also, if the profile runs at an angle, α, with the x-axis then X_j in equation (3.1) needs to be replaced by $X_j\cos\alpha$ (Chakravarthi and Ramamma, 2013b; Chakravarthi et al., 2014).

3.3.1 Effects of offset and strike length on the magnitude of anomaly

Figure 3.2a illustrates the effects of profile offset and strike length of the structure on the magnitude of gravity anomaly produced by a listric fault source, whose geometry is shown in Figure 3.2b. In this case, the detached hanging wall is presumed to consist in two formations namely, 7 km thick-sectioned sediments at the bottom are covered on the top by 3 km thick basalt flow (Figure 3.2b). The assumed densities for the two formations under consideration are 2.85 gm/cm^3 for basalt and 2.3 gm/cm^3 for concealed sediments, respectively (Figure 3.2c). Further, each formation within the hanging wall has a strike length of 50 km (Figure 3.2d).

For such a geologic setting, the anomalies are calculated along two selected profiles PP’ and QQ’ (locations of the profiles are shown in a plan

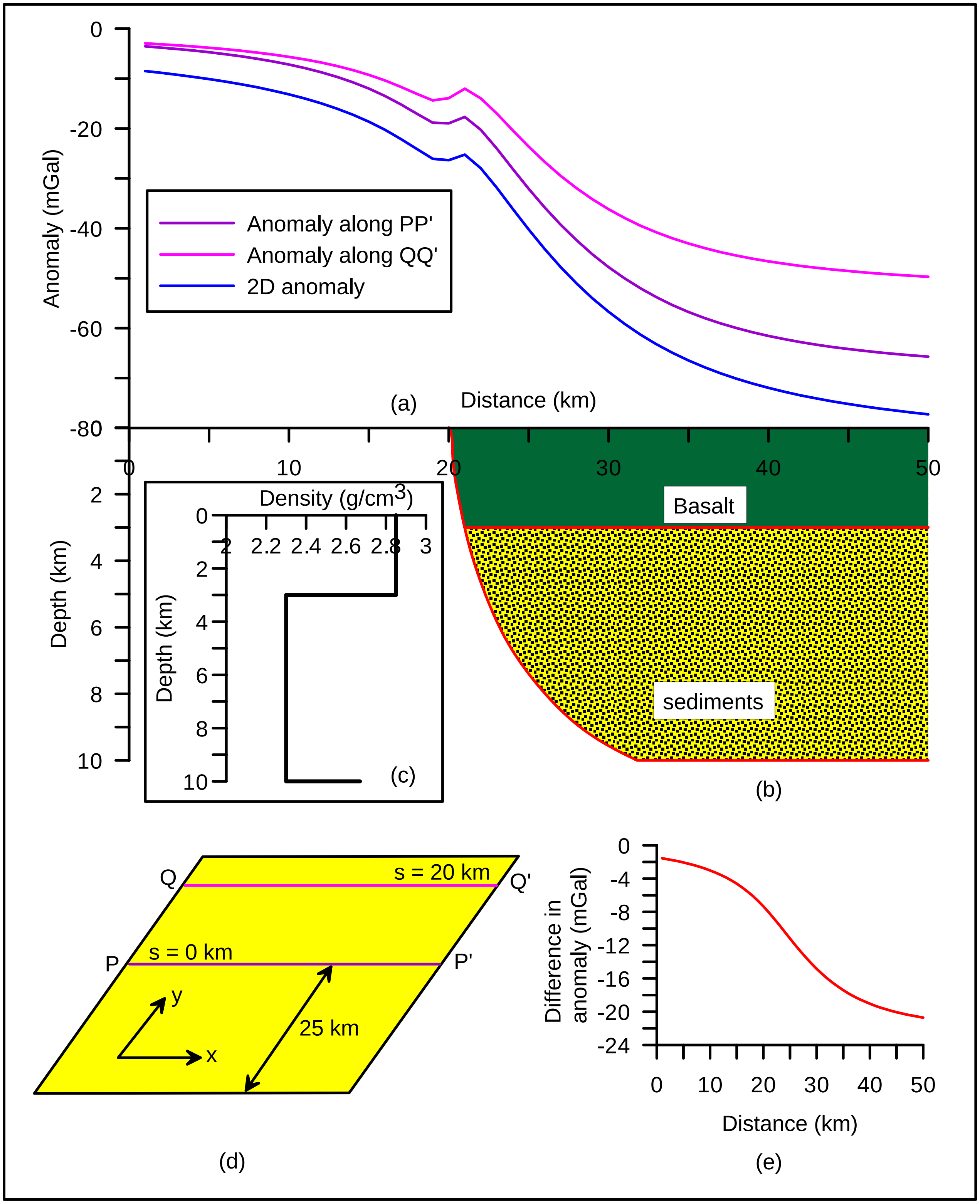


Figure 3.2 (a) Gravity anomalies along two selected profiles, PP' and QQ', across a 2.5D strike listric fault source, whose geometry is shown in (b), (c) density-depth relatioship within the hanging wall, (d) plan view showing the locations of the profiles, (e) differences in magnitudes of gravity anomalies observed along the profiles, PP' and QQ'.

view in Figure 3.2d) at 50 observations in the interval x_j Є [0 km, 50 km] and shown in Figure 3.2a. It is to be noted from Figure 3.2d that the two selected profiles (PP', QQ') run transverse to the strike of the structure but the profile, PP', bisects the strike length whereas the other one, QQ', does not (profile, QQ', is located at an offset of 20 km). One can notice from Figure 3.2a that the gravity anomaly along the profile PP' has portrayed larger a magnitude (absolute) than the corresponding one observed along the QQ' profile.

Furthermore, the anomalies along the two selected profiles closely mimic with each other at the observations resting on the footwall of the structure far away from the fault plane, whereas they depict large deviations over the hanging wall (Figure 3.2a) across the fault plane. The magnitude of the difference between the two anomalies varies from about -1.5 mGal over the footwall to more than -20 mGal above the hanging wall (Figure 3.2e). Therefore, the magnitude of gravity anomaly (absolute) of a 2.5D listric fault source decreases with the increase in offset of the profile, albeit the anomalous source remains the same.

To study the effect of strike length on the magnitude of gravity anomaly, a large value of 1500 km is assigned to the strike of the structure (to consider the structure as 2D) before being performed the forward modeling in the interval x_j Є [0 km, 50 km]. In this case, the other parameters of the structure remain unchanged. The anomalous field of such a 2D structure (shown in

Figure 3.2a as a solid line in blue) deviates significantly from the anomalies produced by the same structure with limited strike length.

Therefore, it is concluded that the gravity anomalies generated by a listric fault source are dependent on both the strike length of the structure and offset of the profile. Hence, these parameters play pivotal roles in the analysis of gravity anomalies of listric fault morphologies.

This chapter deals with the development of an interpretation strategy and related software, FRGMLSTRK, coded in JAVA to calculate as well as to model the gravity anomalies of 2.5D strike-limited listric fault morphologies in real time using a Graphical User Interface (GUI). The advantage of this software is that, besides being platform-independent, it also allows the user to initialize as well as to modify the structure interactively (using simple mouse click operations) until the modeled anomalies fit the observed ones. The applicability of the interpretation technique is demonstrated on a synthetic and a real world gravity anomaly.

3.4 Description of software – FRGMLSTRK

The architecture of the present code, FRGMLSTRK, (Annexure-3A) is based on the MVC pattern shown in Figure 2.2 (Chakravarthi et al., 2014). The 'Model' performs the task of finding the origin of the fault plane from a set of observed gravity anomalies, estimates the coefficients of the prescribed polynomial to describe the fault plane geometry, and computes the modeled gravity anomalies of the structure. The 'View' module executes the task of reading the input data and allows the user to modify the fault plane geometry, depths and densities of several sub-surface formations either independently or in combination and finally displays the results in both ASCII and graphical forms. The ‘Control’ unit executes the task of passing the required actions to the model and view modules. Upon invoking the batch file, the view module appears on the screen as shown in Figure 3.3.

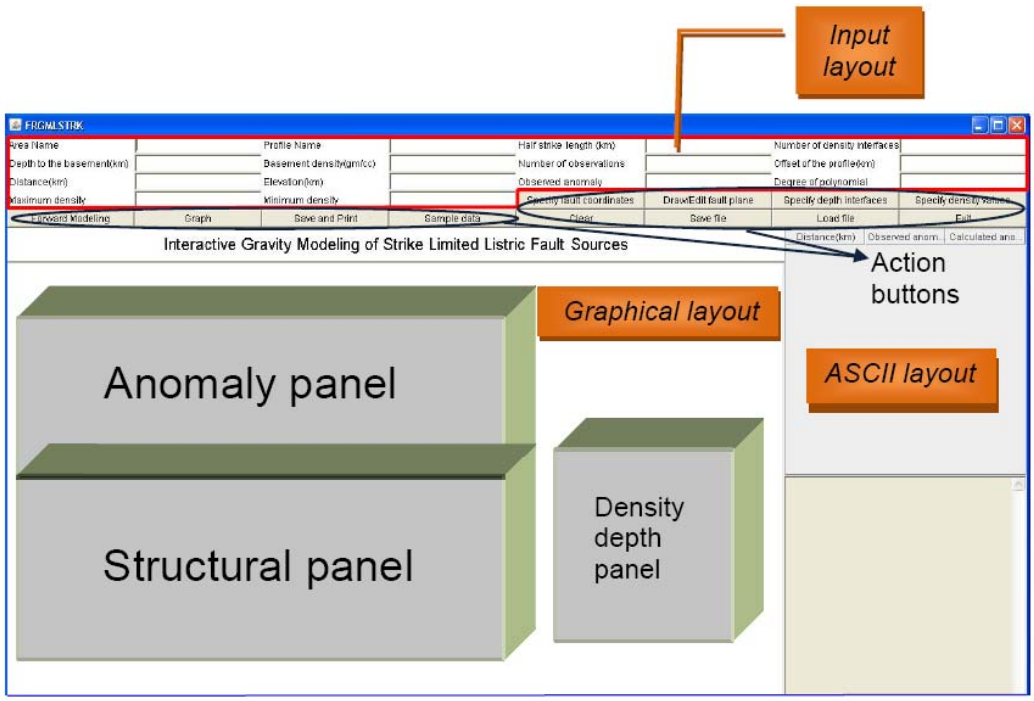


Figure 3.3 View module of FRGMLSTRK

The view module is organized into the input layout, graphical layout and ASCII layout as shown in Figure 3.3. The input layout consists of 14 input fields namely the area name, profile name, half-strike length of the structure in km, number of density interfaces within the hanging wall, depth to basement in km, basement density in gm/cm^3, number of observations, offset of the profile in km, distance and elevation of each observation expressed in km, observed gravity anomaly in mGal, degree of the polynomial to describe the fault plane, minimum and maximum densities of the sub-surface formations in gm/cm^3.

The 12 action buttons (Figure 3.3) of the business logic are: Specify fault co-ordinates, Draw/Edit fault plane, Specify depth interfaces, Specify density values, Forward Modeling, Graph, Save and Print, Sample data, Clear, Save file, Load file, and Exit. Upon entering the input fields and invokes the action button 'Specify fault coordinates' (Figure 3.4), the code automatically identifies the location of the fault plane (shown as 'X' in Figure 3.4) following the methodology described in section 2.4 of Chapter-II. Based on available subsurface information the interpreter selects a few control points in the structural panel (by means of mouse clicks) to construct the geometry of a listric fault plane as shown in Figure 3.4.

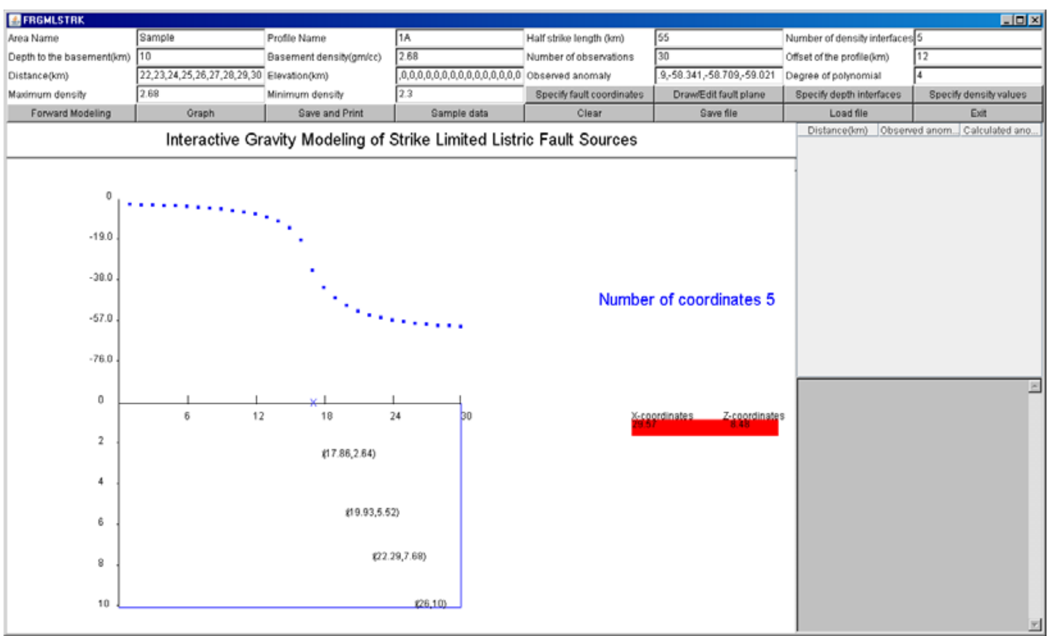


Figure 3.4 Observed gravity anomaly (anomaly panel) and control points to describe fault plane geometry (structural panel).

The number of control points and co-ordinates of each such selected point will be displayed on the right-hand side of the structural panel. If the number of selected control points to describe the fault plane is less than the optimum number (based on the degree of polynomial), then the digit describing the number of points shall be displayed in red. In such a case, the interpreter adds a few more control points in the structural panel till the digit turns to blue (Figure 3.4).

Once the action button 'Draw/Edit fault plane' is invoked (Figure 3.5), the code estimates the coefficients, f_k, by fitting the co-ordinates of the selected control points to a polynomial, $\zeta(z)$. The code uses these estimated coefficients to construct an analytically defined fault plane as shown in Figure 3.5 (structural panel). The same action button can also be used to edit/modify the coordinates of the selected control points (if required) by simple drag and

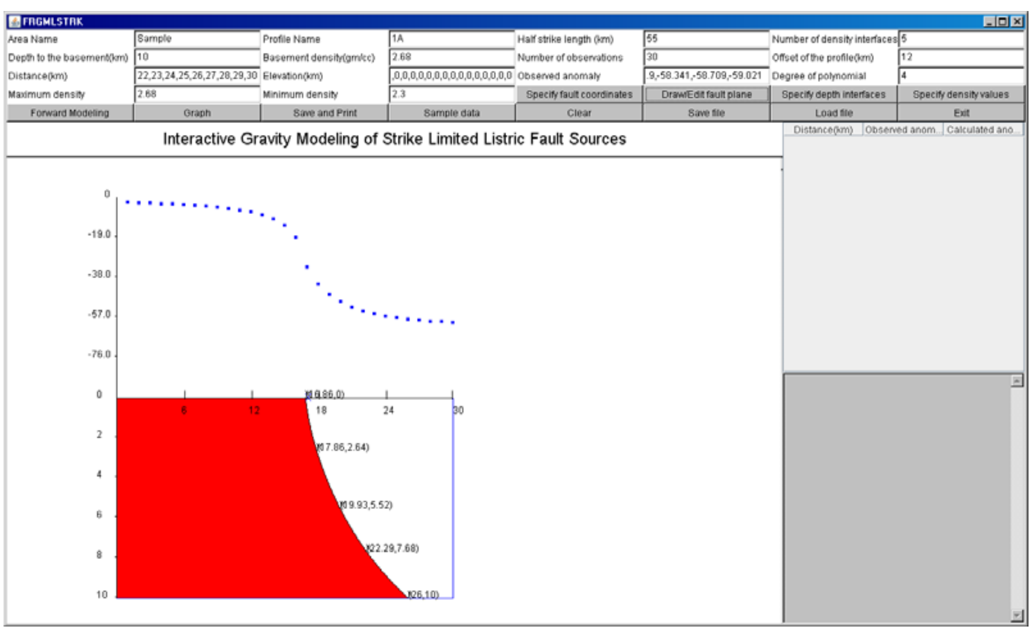


Figure 3.5 Observed gravity anomaly (anomaly panel) and the geometry of an analytically defined fault plane (structural panel)

drop mouse operations. The code then automatically updates the coefficients of the polynomial and reconstructs the modified fault plane.

By invoking the action button ‘Specify depth interfaces’ (Figure 3.6), the user specifies the depths to various density interfaces within the hanging wall of the fault morphology by means of simple mouse clicks in the structural panel (Figure 3.6). The number of density interfaces that are to be specified within the hanging wall shall be displayed on the right-hand side of the anomaly panel. Further, the depth at which a density interface has to be specified can be realized by the mouse operations within the structural panel.

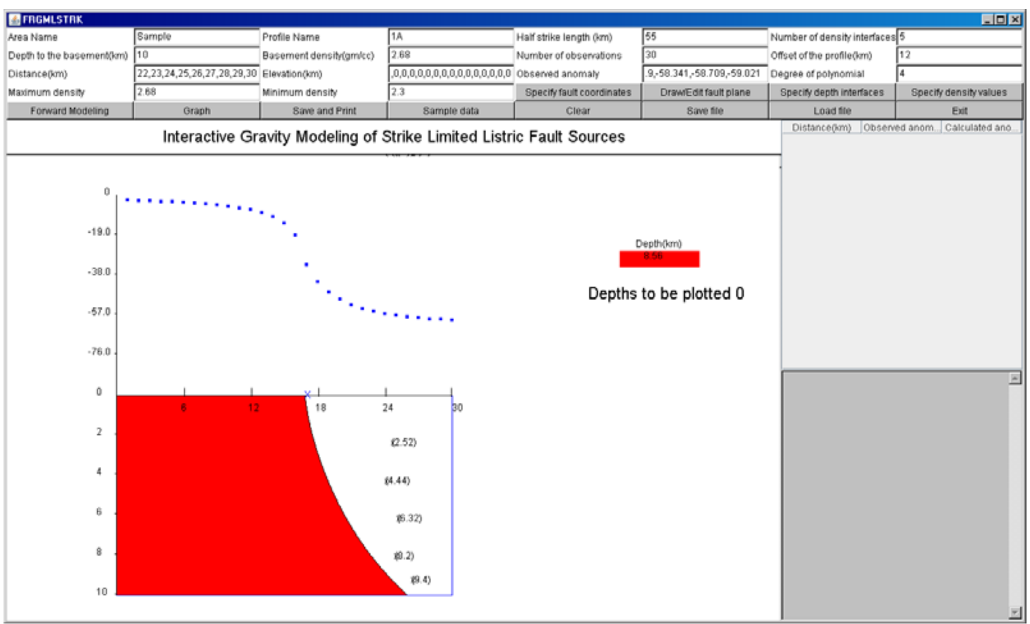


Figure 3.6 Observed gravity anomaly (anomaly panel) and control points selected for specifying the depths to the density interfaces within the structural panel

The interpreter specifies the densities of the sub-surface formations within the hanging wall by invoking the action button ‘Specify density values’ (Figure 3.7). The code then facilitates the user to specify these values by means of mouse clicks in the density-depth panel (Figure 3.7). The number of density parameters to be specified shall be displayed above the density-depth panel.

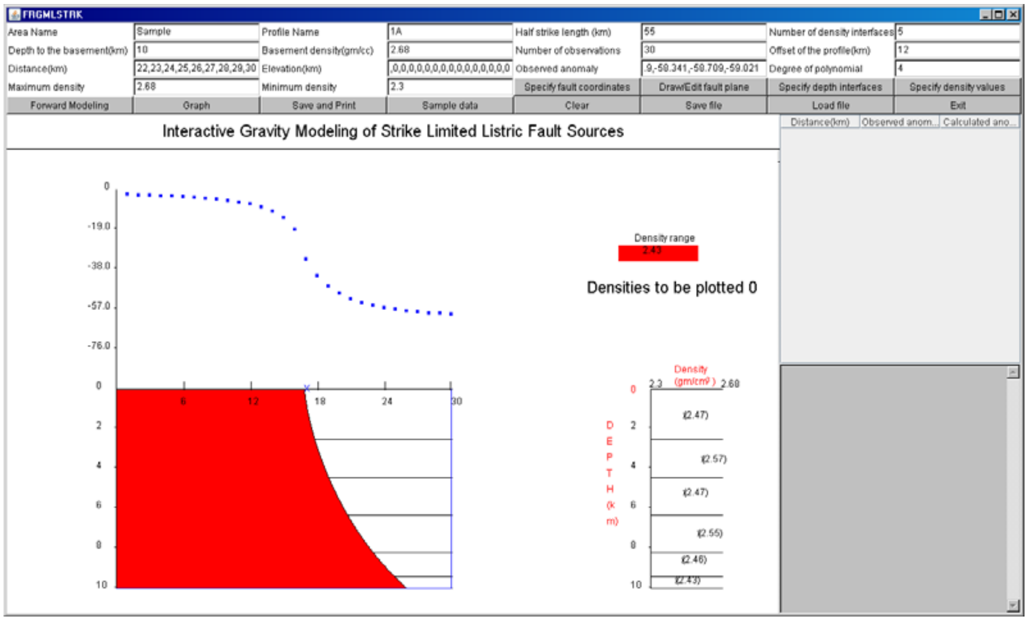


Figure 3.7 Observed gravity anomaly (anomaly panel) and selected control points to specify the densities within the density-depth panel

The code performs the business logic of the algorithm to compute the gravity anomalies of the model space by means of invoking the action button 'Forward modeling' (Figure 3.8). The model response shall be displayed in the anomaly panel (solid line in black in Figure 3.8) along with the observed anomalies (solid dots in blue in Figure 3.8). The estimated coefficients of the polynomial function, density-depth distribution of the sub-surface formations, observed and modeled gravity anomalies are also displayed in ASCII form in the ASCII layout as shown in Figure 3.8.

The differences between the observed and modeled gravity anomalies along the profile can be minimized by changing i) the geometry of the fault plane or ii) densities of formations or iii) depths to density interfaces either independently or in combination.

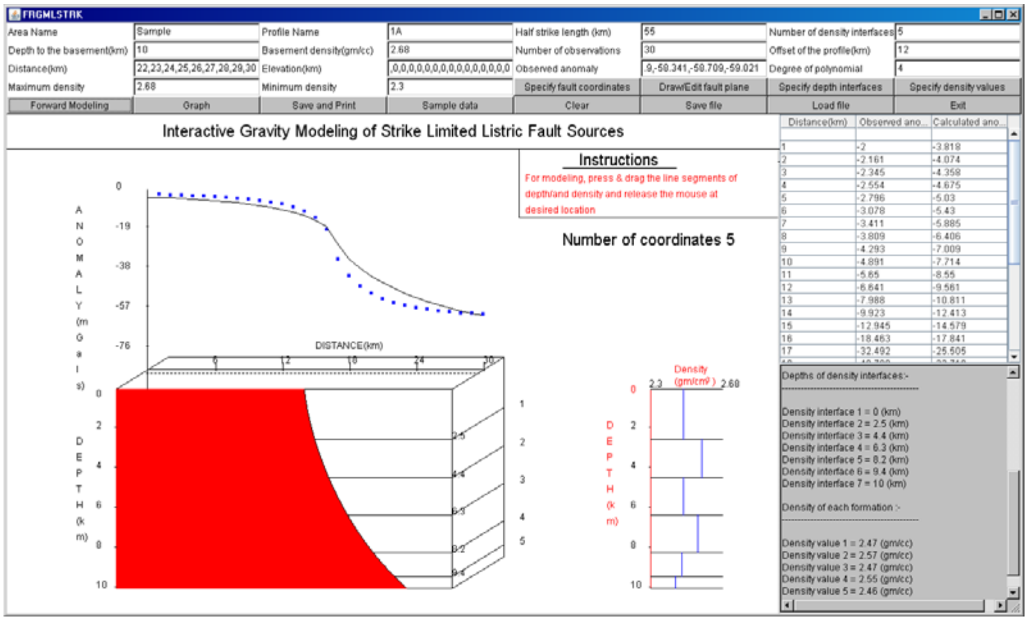


Figure 3.8 Observed and modeled gravity anomalies (anomaly panel), listric fault morphology (structural panel) and density-depth distribution (density-depth panel). The model gravity anomalies are also displayed in ASCII form in the ASCII layout

The geometry of the fault plane can be modified either by invoking 'Draw/Edit fault plane' option (Figure 3.5) or changing the degree of the polynomial. Changes to density and depth parameters can be realized by simple drag and drop mouse operations in the density-depth panel. The model gravity anomalies will be automatically updated and displayed in the anomaly panel along with the observed gravity anomalies (Figure 3.9).

Once the interpreter satisfies with the modeling, the action button 'Graph' shall be invoked to display the sub-surface structure of the hanging wall in colour mode as shown in Figure 3.10.

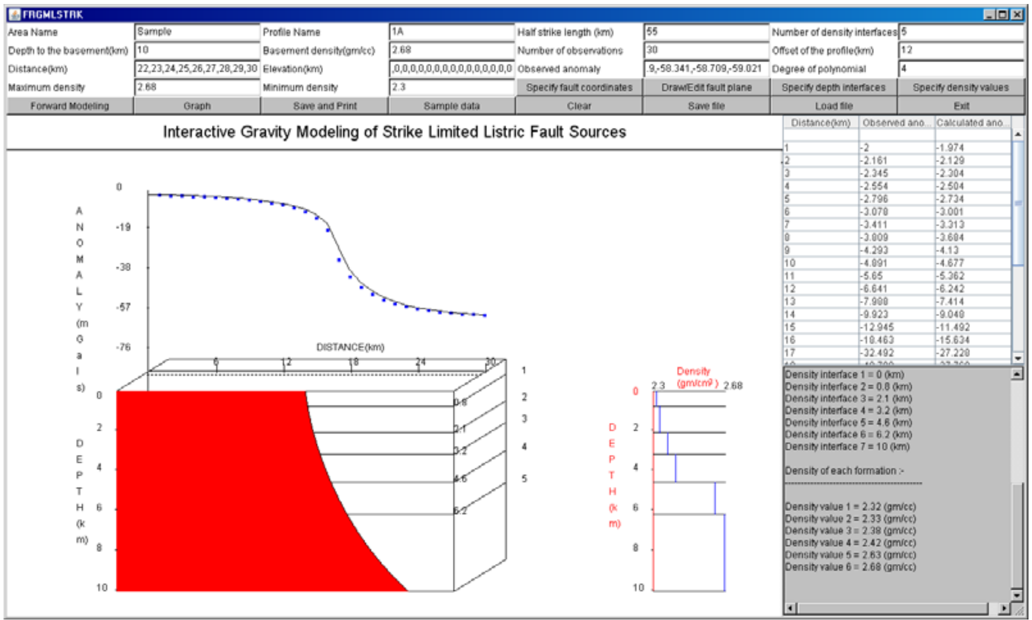


Figure 3.9 Observed and model gravity anomalies (anomaly panel), listric fault morphology (structural panel) with modified density-depth distribution (density-depth panel). The ASCII layout shows the modeled anomalies in ASCII form.

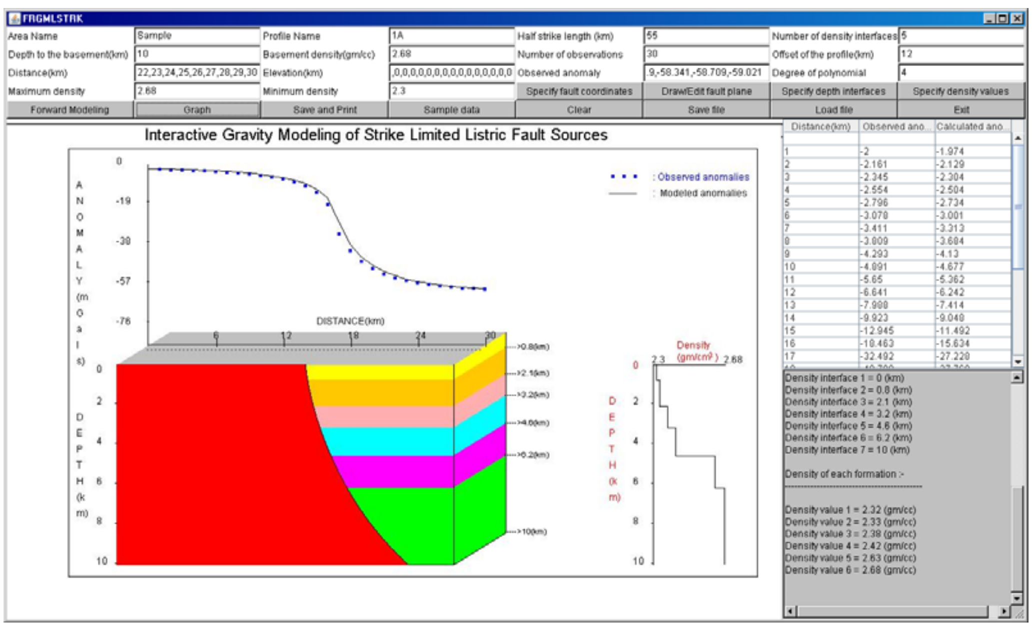


Figure 3.10 Observed and modeled gravity anomalies (anomaly panel), representation of different formations within the hanging wall by different colours (structural panel), and modified density-depth distribution (density-depth panel).

By activating the 'Save and Print' option (Figure 3.11), the interpreter saves the output (interpretation result) in both html and jpg formats at a desired location of the computing machine and opts for printing.

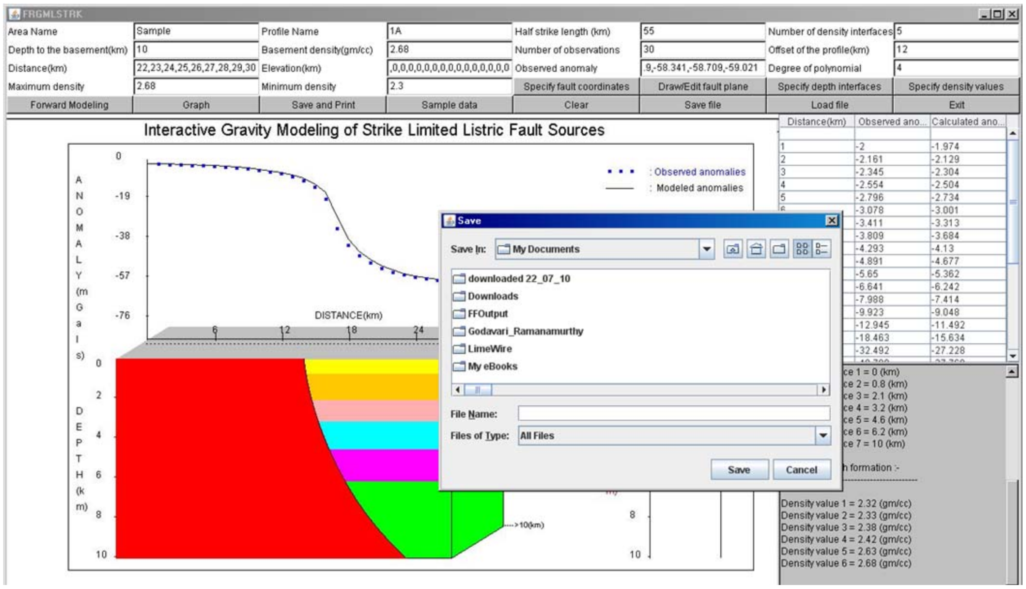


Figure 3.11 Interpreted results in html format

Upon double-click, the html file will be opened in a browser with a print dialog box attached to it. The user selects a printer and press OK to take a printout of the output (Figure 3.12).

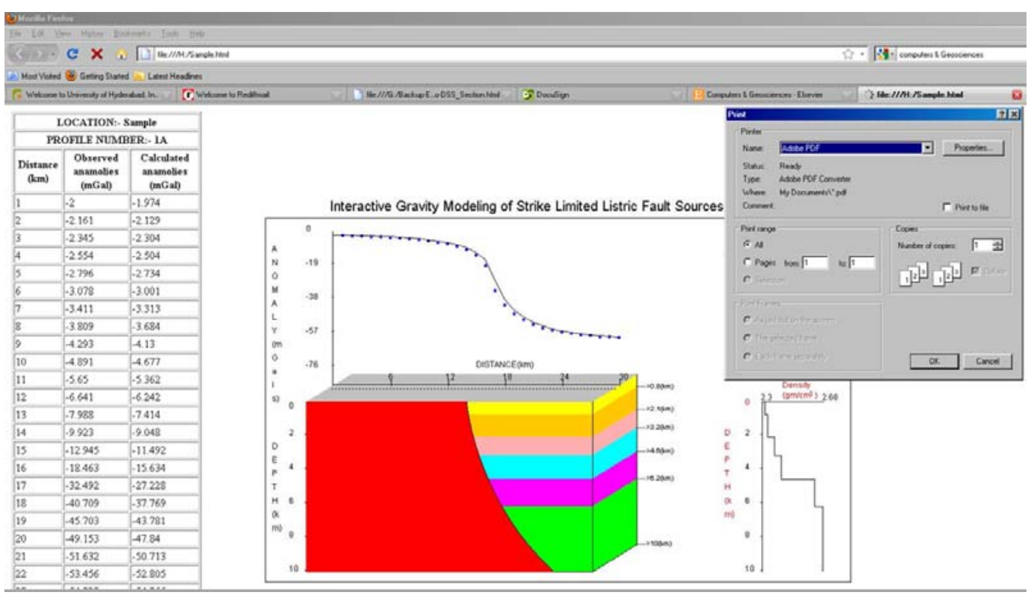


Figure 3.12 Sample output

3.5 Applications

The applicability of the proposed interpretation is demonstrated on a synthetic and a field gravity anomaly across the Aswaraopet master fault, India.

3.5.1 Synthetic example

A set of gravity anomalies produced by a synthetic model of listric fault source (geometry is shown in the structural panel of Figure 3.13) having 40 km half-strike length at 30 km offset in the interval $X_j \in [1, 50 \text{ km}]$ are shown in the anomaly panel of Figure 3.13 as solid dots. In the present case, a 9th degree polynomial defined with a set of ten arbitrarily chosen coefficients (Table 3.1) is used to describe the fault plane geometry (Figure 3.13). The other parameters assumed to generate the gravity anomaly of the structure are: z_T = 0.0 km,

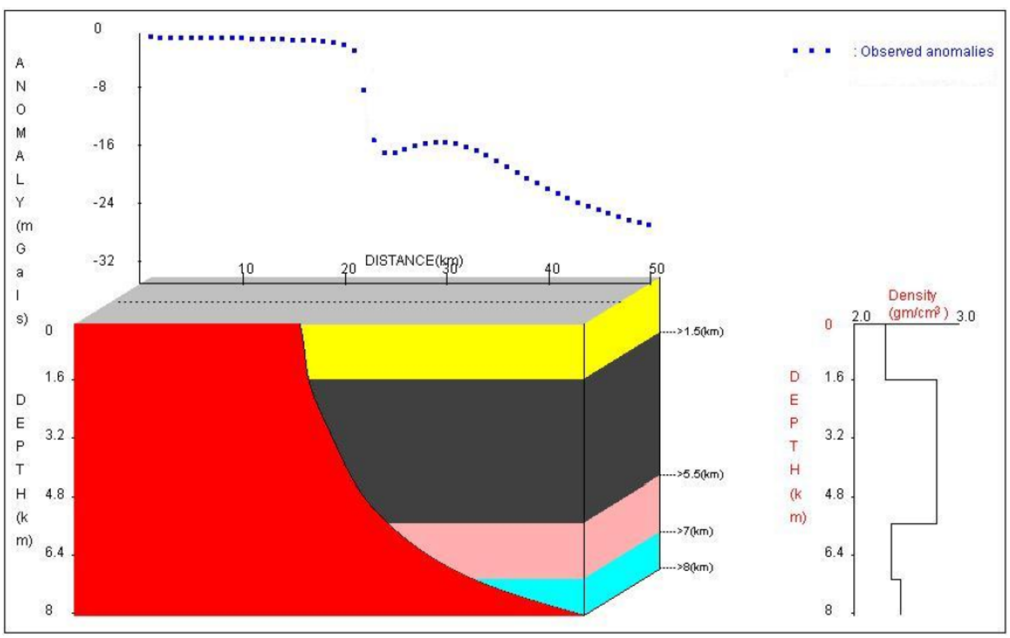


Figure 3.13 Gravity anomalies (anomaly panel) over a four-layered synthetic listric fault morphology (structural panel) with the fault plane described by a 9th degree polynomial. Assumed density-depth data within the hanging wall is shown as a step line in the density-depth panel

Table 3.1
Assumed and estimated coefficients of polynomials, synthetic example

Coefficient	Assumed (9th degree polynomial)	Estimated (3rd degree polynomial)
f_0	22.1804	21.584
f_1	0.6284	1.9776
f_2	0.847	-0.5809
f_3	-2.3276	0.0953
f_4	2.1483	
f_5	-0.9379	
f_6	0.2233	
f_7	-0.0298	
f_8	0.0021	
f_9	-0.0001	

z_B = 8.0 km. The footwall of the structure shown in solid red in the structural panel (Figure 3.13) is assumed to be intact, undeformed and possesses uniform density, whereas the detached hanging wall is presumed to consist in four formations; each one has its own density (density-depth panel of Figure 3.13) and thickness (structural panel of Figure 3.13). The anomaly of such a structure varies from -0.39 mGal over the footwall to -26.21 mGal over the hanging wall with a large gradient in between (anomaly panel of Figure 3.13).

In reality, the absence of additional source(s) of information of the subsurface always makes it difficult to choose appropriate parameters pertaining to the density and thickness of the formations within the hanging wall in addition to the degree of polynomial to describe the fault plane geometry. The present code has the flexibility to incorporate such additional information in the model space. Accordingly, the user may incorporate or change the model parameters either independently or in combination.

To demonstrate the validity of the above statement, the gravity response of the structure (anomaly panel of Figure 3.13) is modeled presuming a 3rd degree polynomial for the fault plane and by (i) changing the depths of the formations alone while keeping the densities intact, and (ii) changing the densities of the formations alone while keeping the thicknesses unchanged. The geometry of the fault plane described by a 3rd degree polynomial with the estimated coefficients given in Table 3.1 is shown in the structural panels of Figures 3.14 and 3.15. Figure 3.14 shows the modeling result of gravity anomalies when the thickness parameters of the formations alone are changed

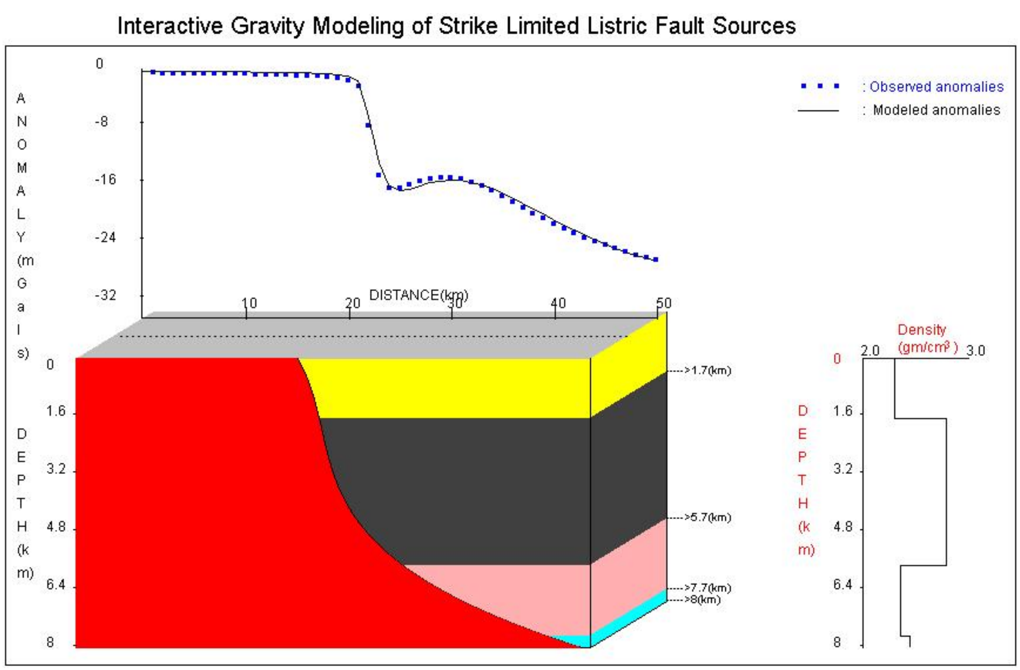


Figure 3.14 Observed and modeled gravity anomalies (anomaly panel) and corresponding modeled listric fault source (structural panel) with the fault plane described by a 3rd degree polynomial. The structure is modeled by modifying the depths of the density interfaces alone.

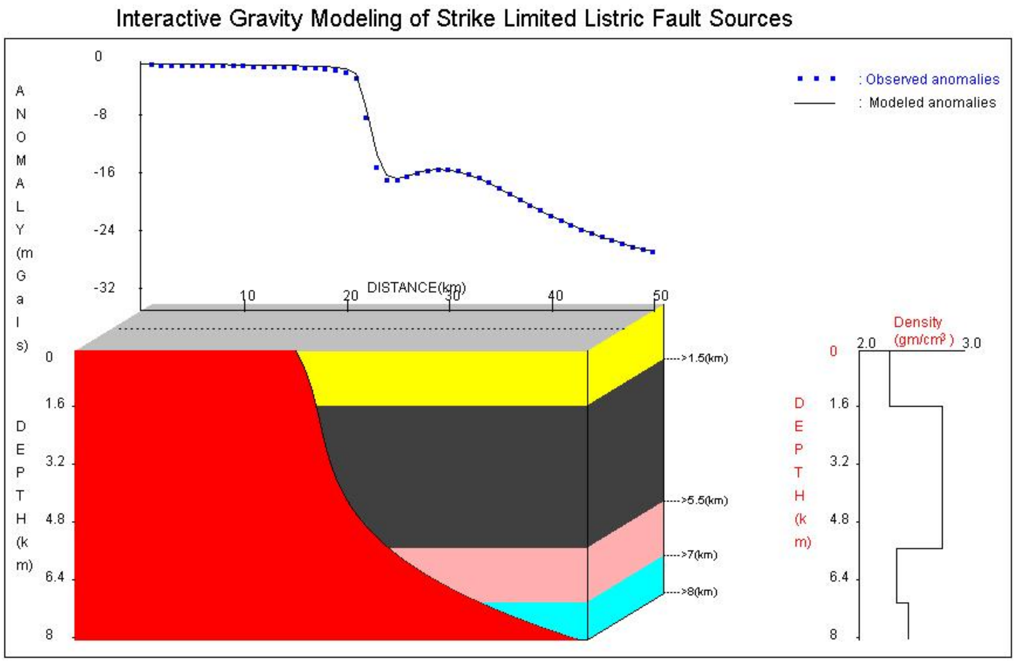


Figure 3.15 Observed and modeled gravity anomalies (anomaly panel) and corresponding modeled listric fault source (structural panel) with the fault plane described by a 3rd degree polynomial. The structure is modeled by modifying the densities of the formations alone.

and Figure 3.15 when densities alone are modified. The modeled gravity anomalies in either case closely fit the observed anomalies. Table 3.2 summarizes the differences between the assumed and estimated parameters of the structure subsequent to modeling in each case.

Table 3.2
Assumed and estimated parameters by interactive modeling
Synthetic example

Formation Number	Assumed depth (km)	Estimated depth (km)	%Error in depth	Assumed density (gm/cm^3)	Estimated density (gm/cm^3)	%Error in density
1	1.5*	1.7**	13.3	2.30*	2.29**	0.4
2	5.5*	5.7**	3.6	2.79*	2.79**	0
3	7.0*	7.7**	10	2.36*	2.36**	0
4	8.0*	8.0**	0	2.44*	2.47**	1.2

* with 7th degree polynomial approximation of the fault plane
** with 3rd degree polynomial approximation of the fault plane

3.5.2 Field Example

The practical applicability of the proposed interpretation is demonstrated with the analysis of gravity anomalies attributable to the Aswaraopet master fault of the Chintalpudi sub-basin in India.

The Chintalpudi sub-basin represents the south-easterly continuation of the Pranhita–Godavari valley. The Archaean gneisses form the basement for the Gondwana sequence within the sub-basin and on the eastern side the basin is bounded by the well-known Aswaraopet master fault, which is exposed to the surface and strikes NNW–SSE over a length of 20 km (Figure 3.16a).

Kaila et al. (1990) have carried out Deep Seismic Sounding (DSS) investigations along a profile across the basin connecting Kallur and Polavaram (Figure 3.16a). The Oil and Natural Gas Corporation Limited (ONGC), India drilled a deep borehole in the basin and encountered the Archaean basement at a depth of 2.935 km (Agarwal, 1995). The density contrast-depth data measured from this borehole of the basin is shown in Figure 3.16b (Chakravarthi, 2003). The gravity data of the basin (Figure 3.16c) has been analyzed by Chakravarthi and Sundararajan (2007) for its basement structure using a 3D inversion.

For the present study, the gravity anomalies of the basin are modeled along a profile, EE', (Figure 3.16a) with an aim to decipher the subsurface structure of the Aswaraopet fault morphology. It is to be noted that the profile, EE', is a part of the DSS profile. To start with, the inferred depth structure of

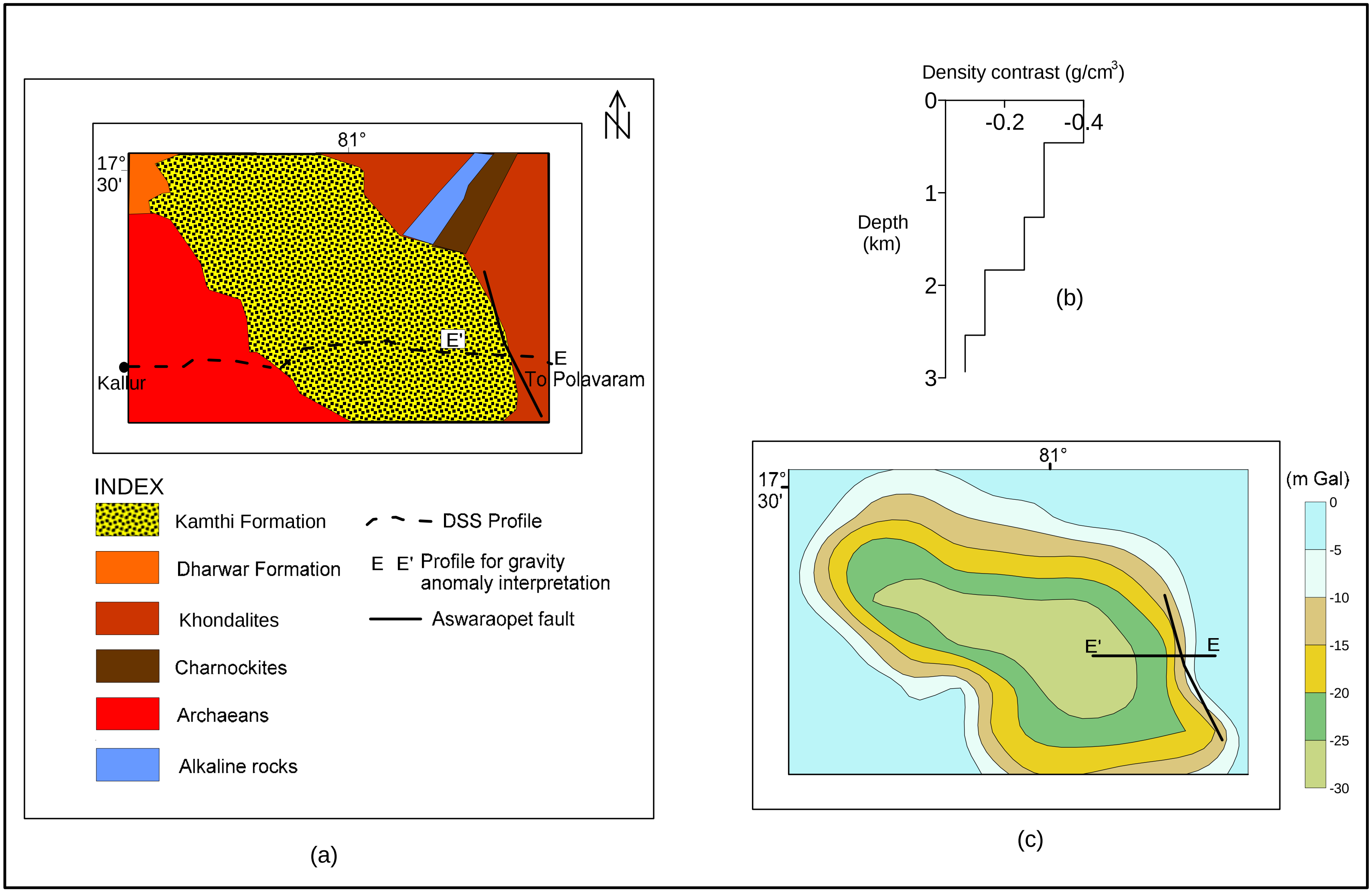


Figure 3.16 (a) Geological map of the Chintalpudi sub-basin, India (modified after Kaila et al., 1990), (b) Measured density contrast depth data, (c) Gravity anomaly map, Chintalpudi sub-basin, India (after Chakravarthi and Sundararajan, 2007)

the Aswaraopet fault derived from DSS investigations (after Kaila et al., 1990) was sampled at 25 closely spaced points in the xz plane. A 6^{th} degree polynomial with a set of 7 coefficients (Table 3.3) was fitted to the coordinates of these selected points to analytically describe the fault plane geometry (structural panel of Figure 3.17). Although, any higher order polynomial can be used to describe the geometry of the inferred fault plane; in the present case, a 6^{th} degree polynomial was found to be adequate.

Table 3.3
Coefficients of 6^{th} degree polynomial,
Aswaraopet fault, Chintalpudi sub-basin, India.

Coefficient	DSS structure	Modeled structure
f_0	21.2107	21.2024
f_1	3.3803	1.3303
f_2	-12.0856	-0.3347
f_3	26.4138	-0.0591
f_4	-18.4317	1.3942
f_5	5.3649	-1.1842
f_6	-0.5545	0.2826

The calculated gravity response of the fault morphology (derived from DSS studies) using the measured density contrast-depth data (Figure 3.16b) is shown in the anomaly panel of Figure 3.17. One can clearly see from Figure 3.17 that the calculated gravity response of the seismically derived structure could not explain the observed gravity anomaly.

Although the differences between the observed and calculated gravity anomalies can be minimized by several means as described in section 3.4; in this case, attempt has been made to modify the geometry of the fault plane alone as the density and depth parameters used in the interpretation are the measured quantities. Accordingly, the fault plane geometry is modified to minimize the errors between the observed and calculated gravity responses (anomaly panel of Figure 3.18). The coefficients of the modified polynomial are given in Table 3.3 and corresponding fault plane geometry in Figure 3.18 respectively (structural panel). The gravity response of the modified structure is shown in the anomaly panel of Figure 3.18.

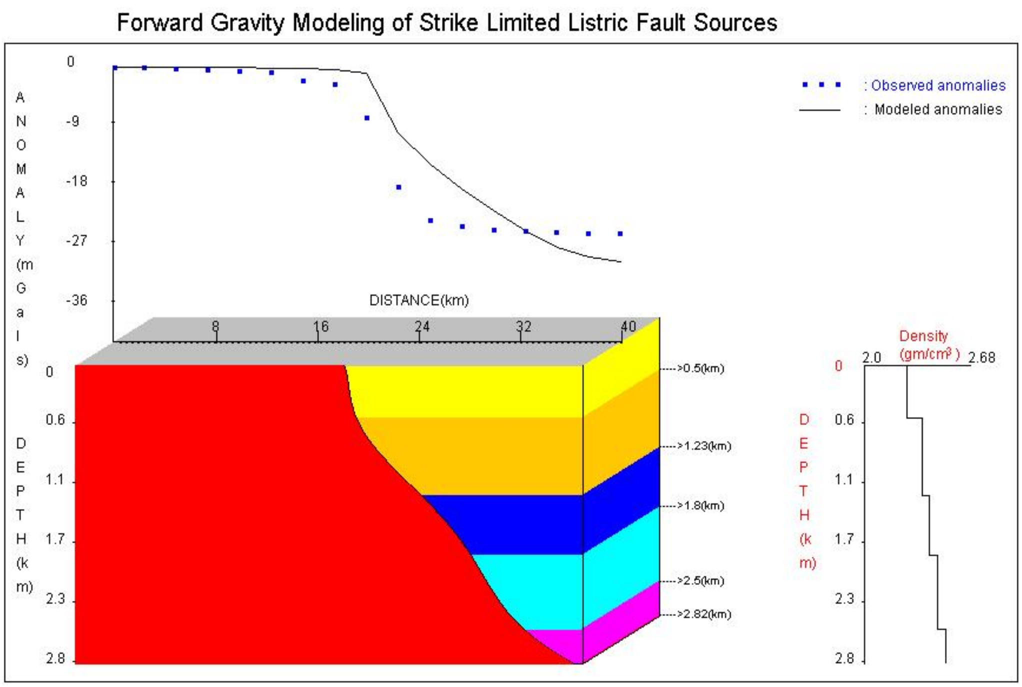


Figure 3.17 Observed gravity anomalies (anomaly panel) across the Aswaraopet master fault, Chintalpudi sub-basin, India. The model geometry shown in the structural panel is based on the DSS investigations (after Kaila et al., 1990). The gravity response of the model is shown by a solid line in the anomaly panel

One can observe from Figure 3.17 that the structure inferred from DSS studies shows high angle dip for the fault plane from the surface to a depth of about 0.6 km, then moderately varying dips up to 1.7 km beyond which it turns again into a high-angle normal fault. On the other hand, the present modeling reveals that the fault plane (structural panel of Figure 3.18), which dips at high angle near the surface, shows similar attitude up to a depth of 1.7 km beyond which it shows moderate dips (Chakravarthi et al., 2014).

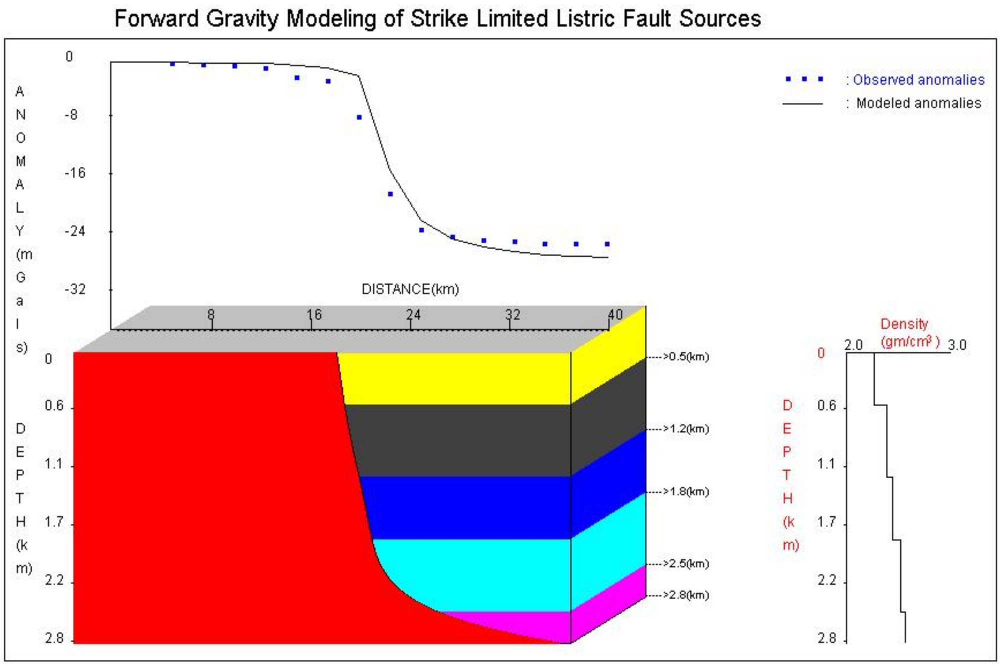


Figure 3.18 Observed and model gravity response (anomaly panel) and modified depth structure (structure panel) from present modeling

One can notice from the anomaly panels of Figure 3.17 and 3.18 that the observed anomalous field varies rather smoothly across the fault plane from the footwall to the hanging wall with a large gradient (4 mGal/km) in between. These characteristics of the observed anomaly are better explained by the interpreted model from the present modeling in comparison to the one proposed by Kaila et al. (1990).

3.6 Results and discussion

A method coupled with GUI based software, FRGMLSTRK, coded in JAVA has been developed to interactively model the gravity anomalies to recover the geometries of 2.5D strike listric fault sources where the detached hanging wall of a structure consists of several geologic formations that have different densities and thicknesses. The software is simple and user friendly in the sense that it allows interactive model construction and modification, the display of fault geometry, depth and the densities of various subsurface formations, and real-time computation of the gravity anomalies arising from the model. Subsequent changes in these parameters, either independently or in combination, can be realized by simple drag and drop mouse operations.

The applicability of the code is demonstrated on a synthetic model and also is exemplified with real-world gravity anomalies arise from the eastern margin of the Chintalpudi sub-basin in India.

In case of synthetic example, the anomalies come up from a typical listric fault morphology that consists of four formations within the hanging wall and the fault plane described by a 9^{th} degree polynomial are analyzed. The analysis showed that different combinations of density and thickness parameters and polynomial functions could equally explain the observed gravity anomalies. Appropriate model can therefore be chosen in light of the information obtained from other geophysical data.

In case of real field example, the observed gravity anomalies across the Aswaraopet master fault of the Chintalpudi sub-basin in India are analyzed using the measured density-depth data from a deep borehole. The depth structure of the fault morphology inferred from the present method explains the observed gravity anomalies better than the model derived from DSS studies.

The advantage of the method and software is that it can be applied to analyze the gravity anomalies of faulted basin margins, even when the profile of interpretation fails to bisect the fault plane.

CHAPTER
FOUR

Simultaneous estimation of multiple density-depth parameters and fault plane geometry from gravity inversion: Application to 2.5D strike listric fault morphologies*

4.1 General:

Interpretation of gravity anomalies of listric fault morphologies amounts to estimating the parameters pertaining to the densities and thicknesses of the formations within the hanging wall, in addition, to decipher the geometry of the fault plane with depth. The interactive modeling technique, described in Chapter-III, is preferable to analyze the gravity anomalies as long as the unknown parameters to be estimated are limited in number. In case, the hanging wall system consists of large number geologic formations having differing densities and thicknesses, it is difficult to decide which parameter/ combination of parameters to be modified appropriately in each scan. Such a constraint often impedes the speed of convergence of the solution. Hence, automatic inversion techniques are preferable to analyze the gravity anomalies of such listric fault sources (Chakravarthi and Pramod kumar, 2015b).

** Published in Geofisica Internacional, (2015, 54, 49-65), Springer.*

This chapter deals with an inversion technique and related GUI based software, GRAVLIS, to analyze the gravity anomalies of listric fault morphologies; where the structure is assumed as i) a strike limited (2.5D) source, and ii) the hanging wall system consists of several geologic formations, each one is having its own density and thickness (Figure 3.1). Based on the differences between the observed and model gravity anomalies, the present inversion estimates the unknown parameters pertaining to either thicknesses or densities of the subsurface formations (because the density and the volume of the source cannot be determined without prior information about one of them) and the coefficients of the predefined polynomial to describe the fault plane geometry (Chakravarthi and Pramod kumar, 2015b). Applicability of the technique is exemplified with the gravity anomalies of a synthetic model and also with the interpretation of real field anomalies of the Aswaraopet mater fault of the Chintalpudi sub-basin, India. The estimated parameters are compared with the assumed parameters in case of synthetic example and with measured density-depth parameters in case of field example.

4.2 Inversion of gravity anomalies

Inversion of gravity anomalies is tantamount to a mathematical exercise of trying to fit the observed gravity anomalies to the anomaly expression to solve the unknown source parameters within specified convergence criteria such that the inferred model is geologically acceptable. Two variants of inversion are developed to analyze the gravity anomalies: i) densities and coefficients of the polynomial, $\zeta(z)$, are estimated while keeping the depths of the density interfaces intact, and ii) depths and coefficients of the polynomial are estimated while keeping the densities of the formations intact. In either case, the interpretation starts by assigning a set of approximate parameters of the structure (densities or depths of the formations) either from known geology or the information supplemented by drilling/other geophysical methods. The proposed algorithm finds approximate location of the fault plane on the profile based on the procedure described in section 2.4 of Chapter-II followed by forward modeling realized through equation (3.1).

Because of the fact that the initial parameters are only approximate, the modeled gravity anomalies, $g_{mod}(X_j, Z_j)$, obtained from equation (3.1) would deviate from the observed anomalies, $g_{obs}(X_j, Z_j)$. The difference between these two anomalies at any point, $P(X_j, Z_j)$, on the profile can be expressed as a cumulative of the products of the gradient of the gravity anomaly with respect each unknown parameter to be estimated and corresponding increment as

$$g_{obs}(X_j, Z_j) - g_{mod}(X_j, Z_j) = \sum_{k=1}^{N} \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_k} da_k + \sum_{m=0}^{N1} \frac{\partial g_{mod}(X_j, Z_j)}{\partial f_m} df_m, \tag{4.1}$$

where, da_k are increments/decrements in the parameters pertaining to either densities or depths, and df_m represents the increments/decrements to the coefficients of the polynomial used to describe the fault plane. Linear expressions similar to equation (4.1) are constructed for each observation on the profile and (N+N1+1) normal equations are formed and solved by minimizing the misfit

$$\sum_{j=1}^{N_{obs}} \left[g_{obs}(X_j, Z_j) - g_{obs}(X_j, Z_j)\right]^2 \qquad (4.2)$$

using Marquardt's algorithm (1970). The relevant system of normal equations is expressed by Chakravarthi and Pramod Kumar (2015b) as

$$\sum_{j}^{N_{obs}} \sum_{m=1}^{N+N1+1} \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_{j'}} \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_m} (1 + \tau_{mj}\delta) da_m$$

$$= \sum_{j=1}^{N_{obs}} Err(X_j, Z_j) \frac{\partial g_{mod}(X_j, Z_j)}{\partial a_{j'}}, j' = 1, 2, \dots, (N + N1 + 1). \qquad (4.3)$$

Here, τ_{mj} = 1, for $m = j$,

= 0, for $m \neq j$.

Here, $a_m = \sigma_m/z_m$ for $m = 1, 2, .., N$ and $a_m = f_{m-(N+1)}$ for $m = N + 1, .., N + N1 + 1$.

The partial derivatives required in the above system of equation (4.3) are obtained numerically by means of calculating the rate of change of gravity anomaly with respect to each unknown parameter to be estimated (Chakravarthi and Pramod Kumar, 2015b). The increments/ decrements, da_m, solved from equation (4.3) are added to/subtracted from the existing parameters, a_m, and the process goes on until one of the termination criteria of the algorithm is fulfilled as detailed in Chapter-II.

4.3 Description of the software – GRAVLIS

Based on the proposed methodology of inversion described in the text, GUI based software named, GRAVLIS, coded in JAVA has been developed to analyze the gravity anomalies produced by 2.5D strike listric fault sources (Annexure 4-A). This software, which works on the MVC pattern (Figure 2.2), can be operated on any platform with preloaded JAVA kit. The module 'Model' performs the task of finding the origin of the fault plane from the observed gravity anomalies, computes the modeled gravity anomalies of the structure, constructs and solves the system of normal equations besides performing the business logic of the algorithm. The 'View' module reads the input data and displays the animated versions of the inversion process, whereas the ASCII layout provides the interpreted results in ASCII form. The role of 'Control' unit is to pass on the required actions to the model and view modules as and when necessary.

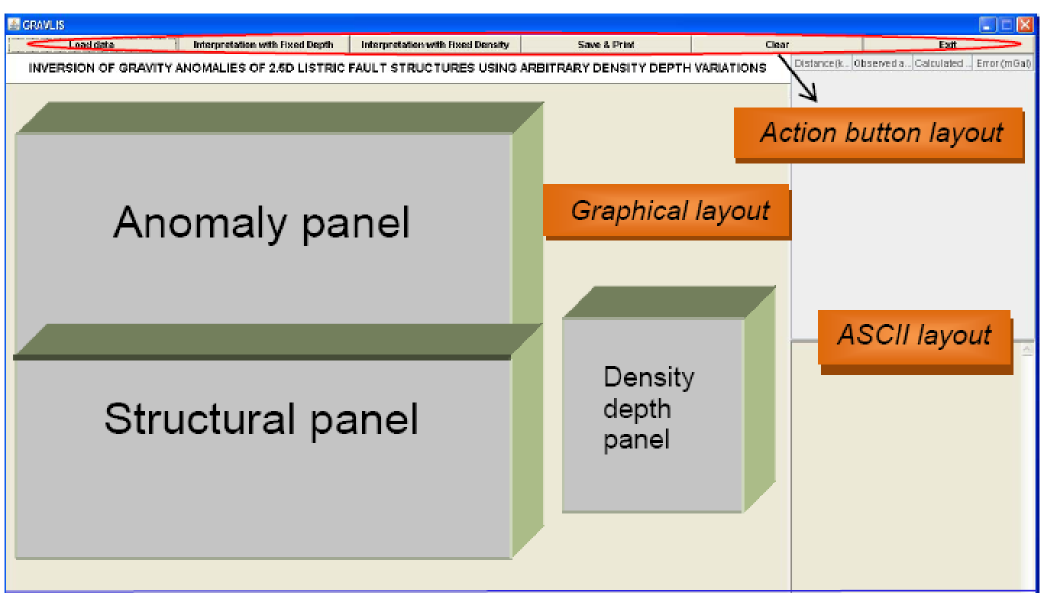


Figure 4.1 View module of GRAVLIS

The view module (Figure 4.1), which appears on the screen upon invoking the batch file, is further organized into the action button layout on the top, graphical layout in the bottom and ASCII layout in the right. The input to the code consists of the area name, profile ID, number of observations, distance and elevation of each observation expressed in km, observed anomaly in mGal, half-strike of the structure in km, offset of the profile in km, number of formations within the hanging wall, basement and formations densities in gm/cm^3, initial/approximate depths of interfaces in km (in case of depths and fault plane inversion) or initial/approximate densities of formations in gm/cm^3 (in case of density and fault plane inversion), number of iterations to be performed and the allowable error.

The user enters the enlisted input parameters in a Microsoft Excel sheet and reads the file to the software by means of 'Load data' action button (Figure 4.1). The proposed variants of inversion (described in section 4.2) can be performed on the input data by selecting either 'Interpretation with Fixed Depth' or 'Interpretation with Fixed Density' option from the action button layout (Figure 4.1).

During the process of inversion, this software displays i) the changes in fault plane geometry, ii) changes in thicknesses/densities of the formations (depending upon the selected option), and iii) nature of fit between the observed and modeled gravity anomalies in animated forms in respective panels. The 'Save and Print' action button enables the user to save the interpreted results in both html and jpg formats and allows for printing.

4.4 Applications

The proposed inversion is applied over two anomaly profiles to demonstrate its validity and applicability; one synthetic and the other real.

4.4.1 Synthetic Example

Figure 4.2a shows a set of noisy gravity anomalies attributable to a synthetic model of 2.5D strike listric fault source, whose geometry is shown in Figure 4.2b. The structure has a half strike length of 50 km. The anomalies (shown as solid dots in Figure 4.2a) are produced at zero offset on a horizontal plane in the interval x_j Є [0, 80 km]. In this case, the pseudorandom noise was Gaussian, with zero mean and a standard deviation of 0.14 mGal. The footwall remains intact and undisturbed, whereas the detached hanging wall consists in four formations namely: a massive basalt of 3.5 km thick at the top is followed successively by 1.5 km thick sediments, 3 km thick vesicular basalt and 2.0 km thick compacted sediments above the basement. The assumed densities of four formations under consideration are given in Table 4.1 and shown in Figure 4.2c.

Table 4.1
Assumed and estimated densities in case of synthetic example

Formation	Assumed density (gm/cm^3)	Initial density (gm/cm^3)	Estimated density (gm/cm^3)	Error (%)
Compact basalt	2.9	2.0	2.9	0.0
Sediments	2.4	2.0	2.44	1.67
Vesicular basalt	2.8	2.0	2.78	0.71
Compacted sediments	2.5	2.0	2.51	0.4

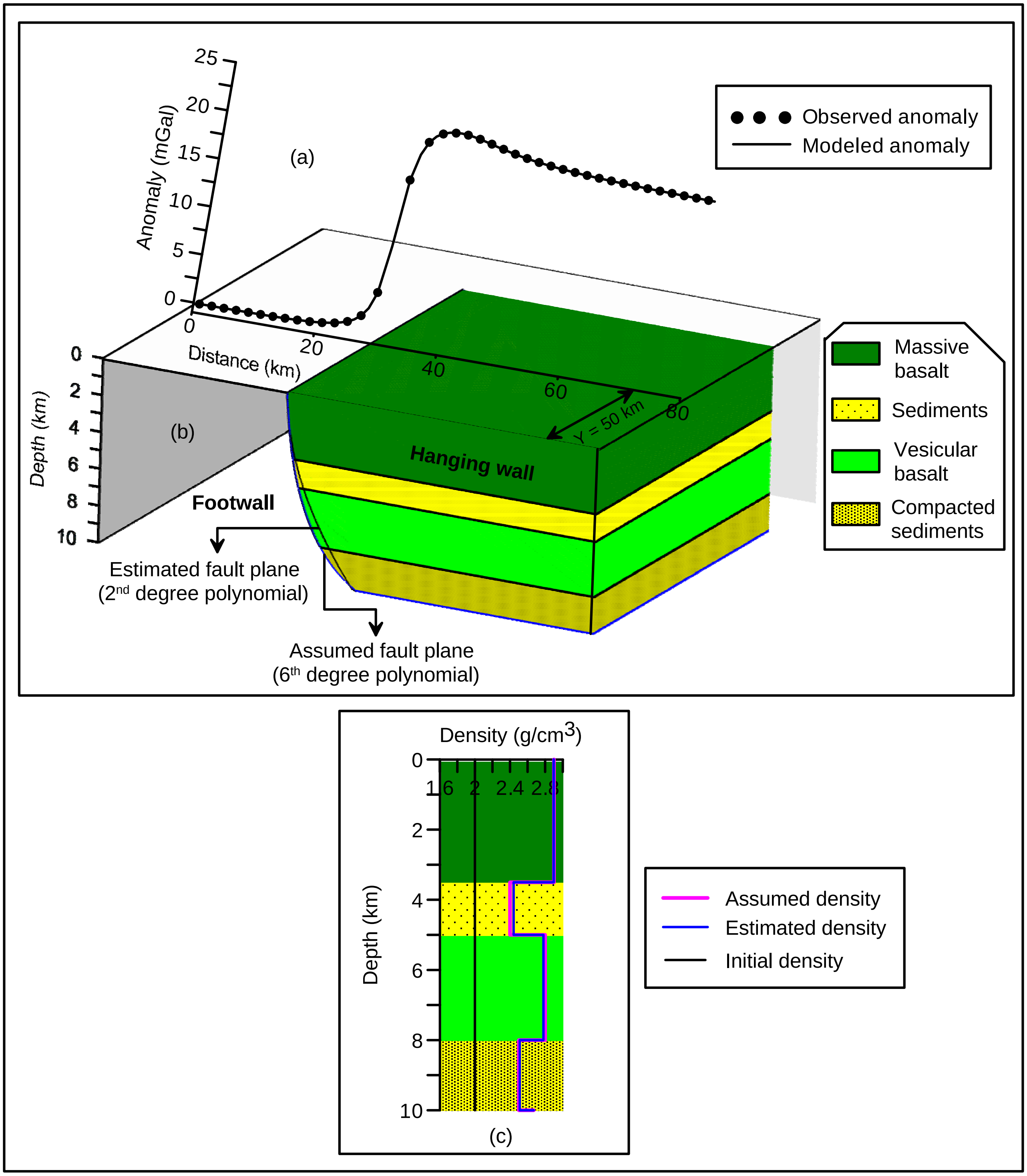


Figure 4.2 (a) Observed and modeled noisy gravity anomalies, (b) four layered hanging wall system of a synthetic listric fault source with assumed and modeled fault planes described by 6th and 2nd degree polynomials, (c) assumed, initial and modeled densities. Depths of density interfaces are fixed during the inversion

In this case, the density of the undisturbed footwall is assumed as 2.67 gm/cm^3. Further, a 6th degree polynomial with a set of seven arbitrarily chosen coefficients (Table 4.2) is used to describe the geometry of the fault plane (shown as solid line in blue in Figure 4.2b).

Table 4.2
Assumed and estimated coefficients of the polynomial, $\zeta(z)$, synthetic example

Coefficient	Assumed coefficients of 6th degree polynomial	Estimated coefficients of the 2nd degree polynomial in case of density and fault plane inversion	Estimated coefficients of the 2nd degree polynomial in case of depth and fault plane inversion
f_0	30.0144	30.0346	30.0299
f_1	0.0965	0.0729	0.0765
f_2	0.1845	0.1059	0.1102
f_3	-0.0732		
f_4	0.0171		
f_5	-0.0017		
f_6	7E-005		

Treating the noisy anomalies shown in Figure 4.2a (solid dots) as the observed ones, inversion was carried out in two prong strategies as discussed in section 4.2. In either case, a 2nd degree polynomial was used in the inversion to simulate the fault plane geometry, in place of a 6th degree, in order to study its effect on the interpretation, if any.

4.4.1.1 Simultaneous estimation of densities and fault plane geometry

Initially a value of 2.0 gm/cm^3 has been assigned to the density of each subsurface formation (Table 4.1 and Figure 4.2c) prior to perform the inversion on noisy anomalies. The algorithm calculates the density contrast of each subsurface formation with respect to the footwall and uses them to compute the gravity effect of the structure. The approximate location of the fault plane identified by the algorithm is at 30.07 km on the profile. The code had performed 69 iterations before it got terminated as the resulting misfit (equation 4.2) fell below a predefined allowable error of 0.017 mGal. No significant improvements in either densities or coefficients of the polynomial are observed beyond the concluding iteration (Figure 4.3b).

The modeled gravity anomalies (shown as a solid line in Figure 4.2a) at the end of the 69th iteration closely fit the observed ones. A maximum error of 0.044 mGal between the observed and modeled gravity anomalies is observed exactly at the 36th km on the profile (Figure 4.3a). The value of misfit had reduced drastically from its initial value of 3550019 to 1.43 at the end of the 34th iteration and then gradually reaches to 0.016 at the end of the concluding iteration (Figure 4.3b). The estimated parameters pertaining to the densities and coefficients of the 2nd degree polynomial from the inversion are given Table 4.1 and Table 4.2 and shown graphically in Figure 4.2c and 4.2b respectively. The percentage of error between the assumed and estimated densities is given in Table 4.1. Furthermore, the changes took place in each estimated parameter (densities and coefficients of the polynomial) against the iteration number are illustrated in Figure 4.3b. It is noticed from Figure 4.2b that the modeled fault plane by a 2nd degree polynomial deviates only marginally from the assumed fault plane described with a 6th degree polynomial.

The predicted density (Table 4.1 and Figure 4.2c) of the bottom most sedimentary pulse at depth is marginally overestimated, whereas the densities

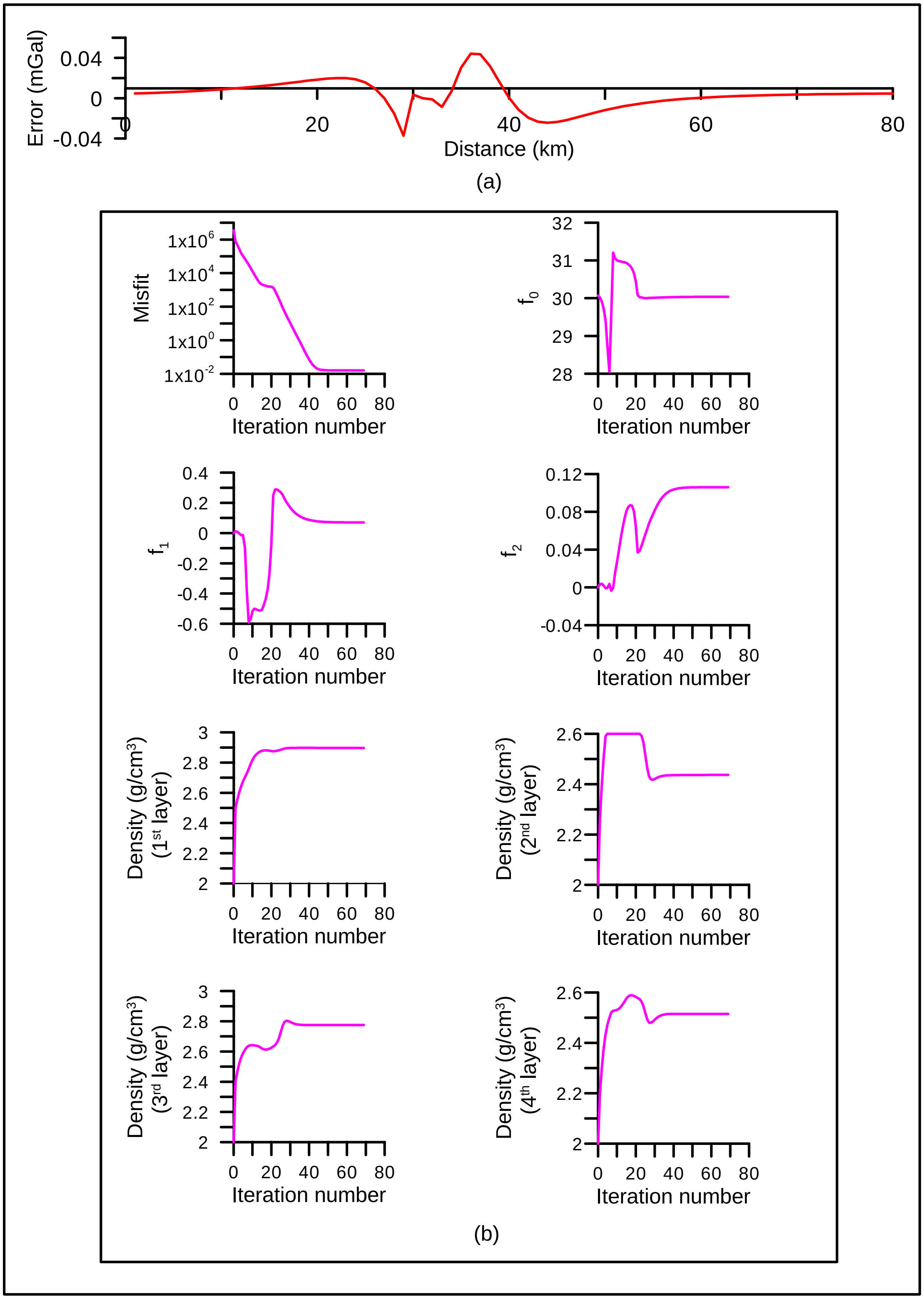


Figure 4.3 (a) Error analysis between the observed and modeled gravity anomalies, (b) changes in misfit, coefficients of a 2nd degree polynomial, and densities of subsurface formations against the iteration number.

of compact and vesicular basalts are slightly underestimated. Such errors between assumed and estimated densities are acceptable considering the presence of significant level of pseudorandom noise in the anomalies produced by the structure. Therefore, the fault plane whether it is described by a 2^{nd} degree or a 6^{th} degree polynomial does not appreciably affect the fault plane geometry and the estimated densities of the structure.

The sample output generated by GRAVLIS in html format is shown in Annexure 4-B.

4.4.1.2 Simultaneous estimation of depths and fault plane geometry

The inversion process is repeated on the noisy gravity anomalies to estimate the depths of four concealed density interfaces and three coefficients of the polynomial by keeping the density parameters unchanged. In this case, the initial depths assigned to four density interfaces are given in Table 4.3 and shown in Figure 4.4c (dotted lines).

One can notice from Table 4.3 that the initial depths of density interfaces are significantly differ from the assumed/true model parameters. As in the previous case, the approximate location of the fault plane identified by the algorithm at 30.07 km was assigned to the first coefficient of the polynomial, whereas other coefficients are set to zero.

For such an inversion, the code took 45 iterations before it got terminated. The misfit (equation 4.2) had reduced from its initial value of

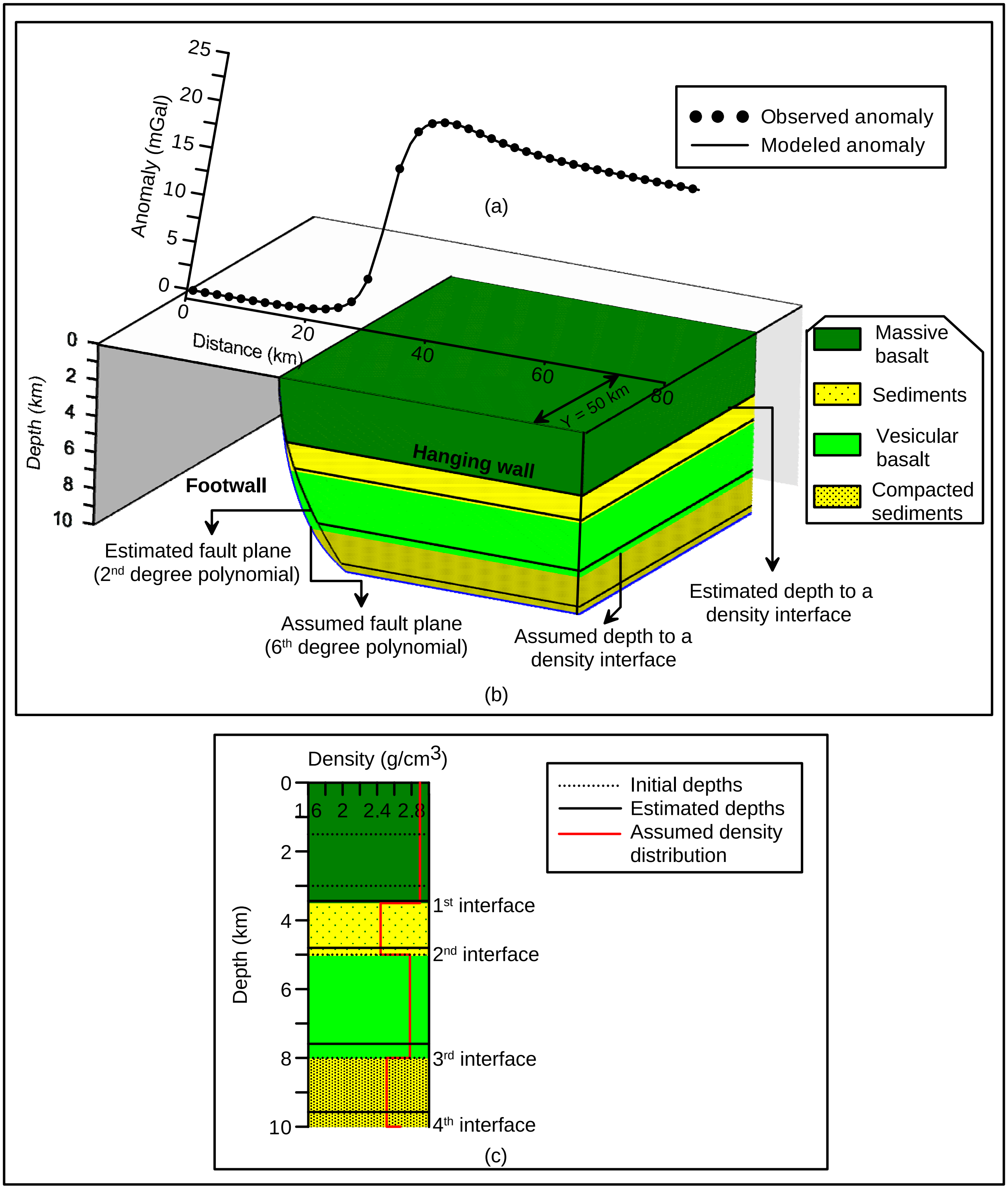


Figure 4.4 (a) Observed and modeled noisy gravity anomalies, (b) four layered hanging wall system of a synthetic listric fault source with assumed and modeled fault planes described by 6th and 2nd degree polynomials, (c) assumed and estimated depths to density interfaces. Densities remain unchanged during the inversion

45565.3 for the starting model to 0.7 at the end of the 19th iteration and then slowly attained to 0.004 at the end of the 45th iteration (Figure 4.5b). No appreciable changes in estimated depths and coefficients of the polynomial are found beyond the concluding iteration (Figure 4.5b).

Table 4.3
Assumed and estimated depths of density interfaces, synthetic example

Interface	Assumed depth (km)	Initial depth (km)	Estimated depth (km)	Error (%)
Compact basalt/sediments	3.5	1.5	3.44	1.71
Sediments/ Vesicular basalt	5.0	3	4.8	4.0
Vesicular basalt/compact sediments	8.0	5	7.6	5.0
Compact sediments/basement	10.0	8	9.57	4.3

The fit between the observed and modeled gravity anomalies at the end of the 45th iteration is satisfactory (Figure 4.4a). The estimated depths to four density interfaces are given in Table 4.3 and shown graphically in Figures 4.4b and 4.4c (solid lines). The estimated coefficients of the 2nd degree polynomial to describe the fault plane are given in Table 4.2 and shown in Figure 4.4b. By and large, the modeled fault plane (simulated by a 2nd degree polynomial) closely mimics the assumed one described by a 6th degree polynomial (Figure 4.4b). In this case, a maximum error of -0.021 mGal between the observed and modeled gravity anomalies is observed at the 40th km on the profile (Figure 4.5a). The changes in the estimated parameters (depths to density interfaces

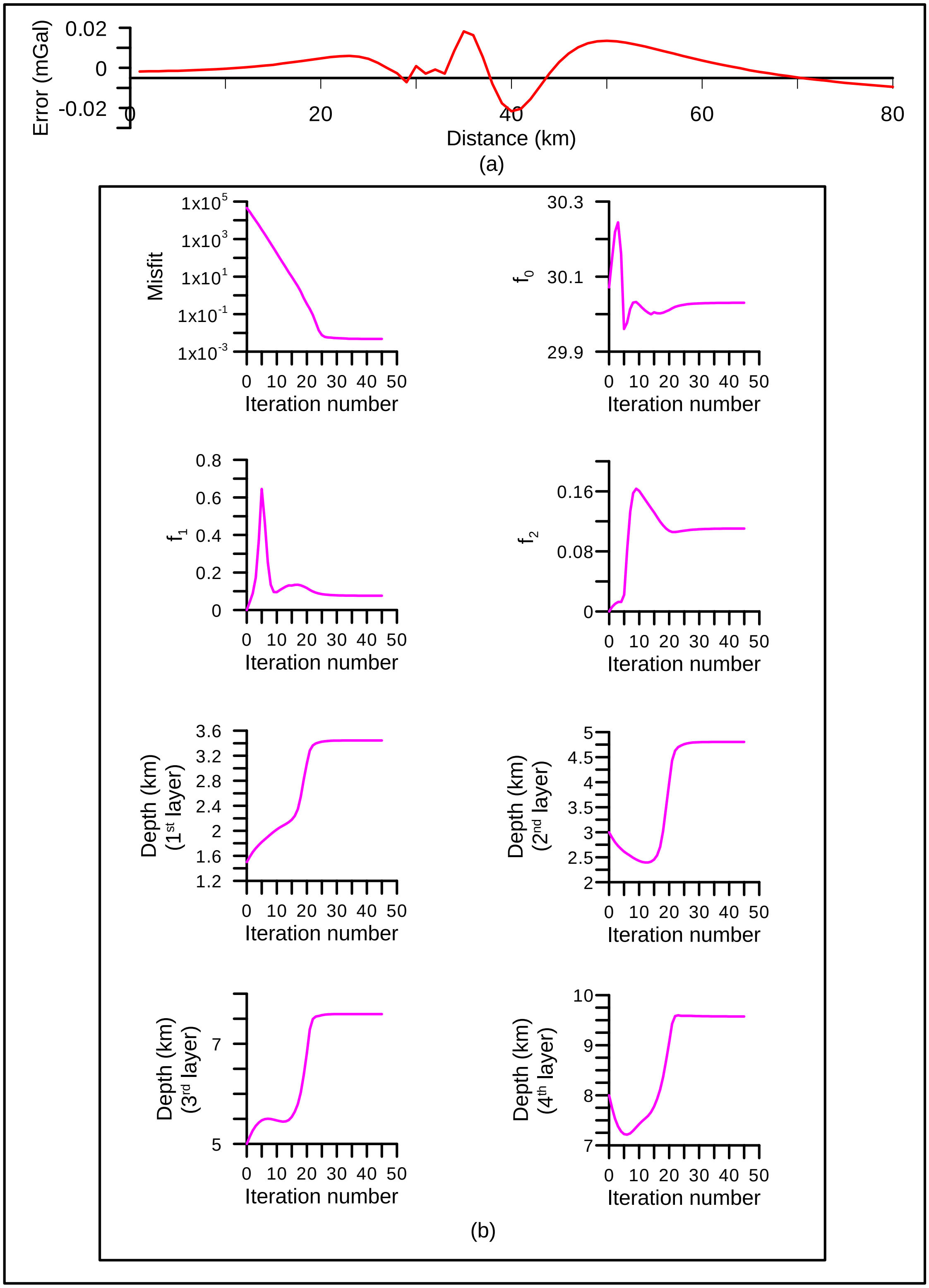


Figure 4.5 (a) Error analysis between the observed and modeled gravity anomalies, (b) changes in misfit, coefficients of a 2nd degree polynomial, and depths of various density interfaces against the iteration number.

and coefficients of the 2nd degree polynomial) against the iteration number are illustrated in Figure 4.5b.

It is to be noted from Table 4.3 and Figure 4.4c that the estimated depths to the four density interfaces are marginally underestimated, with a maximum error of 5% found at the interface between vesicular and compact sediments at depth. However, these errors between the assumed and estimated parameters are insignificant considering the fact that the observed anomalies of the structure are noisy.

In short, the fault plane whether it is described by a 2nd degree or a 6th degree does not appreciably affect the estimated densities or depths of the formations within the hanging wall of the structure. However, the choice of a 2nd degree polynomial in the inversion has resulted in slightly underestimating the amount of extension when the anomalies are analyzed to estimate the depths of the density interfaces (Chakravarthi and Pramod Kumar, 2015b).

4.4.2 Field Example

The proposed inversion technique is applied to analyze the observed gravity anomalies (shown in the anomaly panel of Figure 3.17) across the Aswaraopet master fault of the Chintalpudi sub-basin in India. It is to note that the same profile was interpreted in section 3.5.2 of Chapter-III by interactive modeling. In this case, the anomalies are scaled with reference to an observation in the vicinity of the exposed fault trace. A total of 29 gravity anomalies spread over a profile length of 24 km across the fault trace are

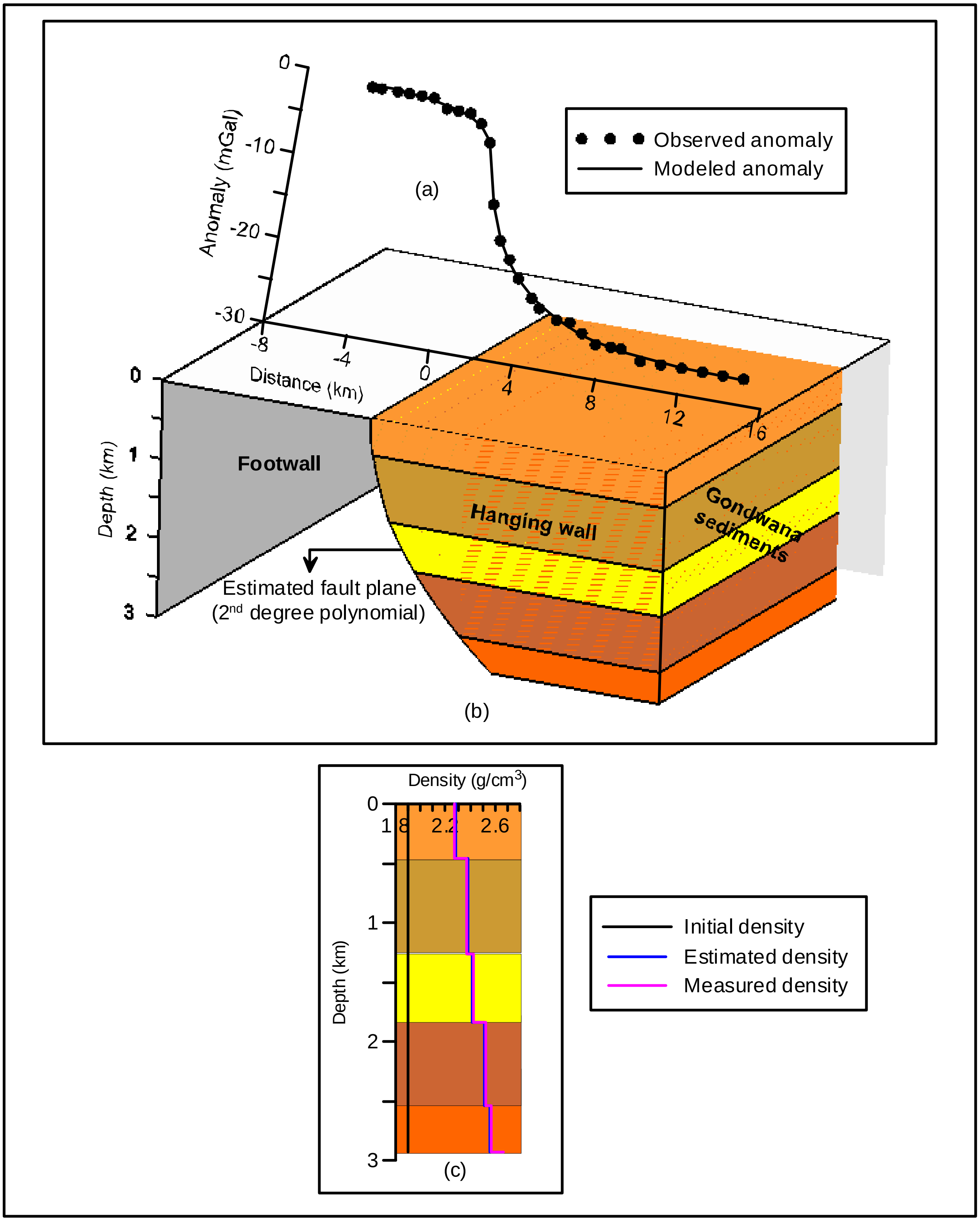


Figure 4.6 (a) Observed and modeled gravity anomalies, (b) inferred geometry of the Aswaraopet master fault morphology, Chintalpudi subbasin, India, (c) measured, initial and estimated densities. Depths of density interfaces are fixed during the inversion

considered for the inversion. The gravity anomalies attain minimum and maximum magnitudes (absolute) over the footwall and hanging wall systems of the fault structure and hence the choice of considering a profile length of 24 km across the fault ramp is justified.

As in the case of synthetic example, discussed in Chapter 4.4.1, the observed anomalies are analyzed in two ways. In either case, a 2nd degree polynomial is opted in the inversion to simulate the geometry of the fault plane. The initial/approximate parameters pertaining to densities (in case the inversion is performed for estimating the densities and polynomial coefficients) and depths (in case the inversion is performed for estimating the depths and polynomial coefficients) are given in Table 4.4 and Table 4.5 and graphically shown in Figure 4.6c and 4.8c (solid lines in black) respectively.

Table 4.4
Measured and estimated densities, Chintalpudi subbasin, India.

Formation	**Measured density (gm/cm^3)**	**Initial density (gm/cm^3)**	**Estimated density (gm/cm^3)**	**Error (%)**
1	2.27	1.9	2.279	0.396
2	2.37	1.9	2.380	0.422
3	2.42	1.9	2.410	0.410
4	2.52	1.9	2.517	0.119
5	2.57	1.9	2.562	0.311

Although, the measured density-depth data of the basin is known apriori, the inversion was performed on the observed anomalies to examine whether the estimated parameters from the inversion mimic the measured ones or not. It can be seen from Tables 4.4 and 4.5 and Figures 4.6c and 4.8c that the initial/

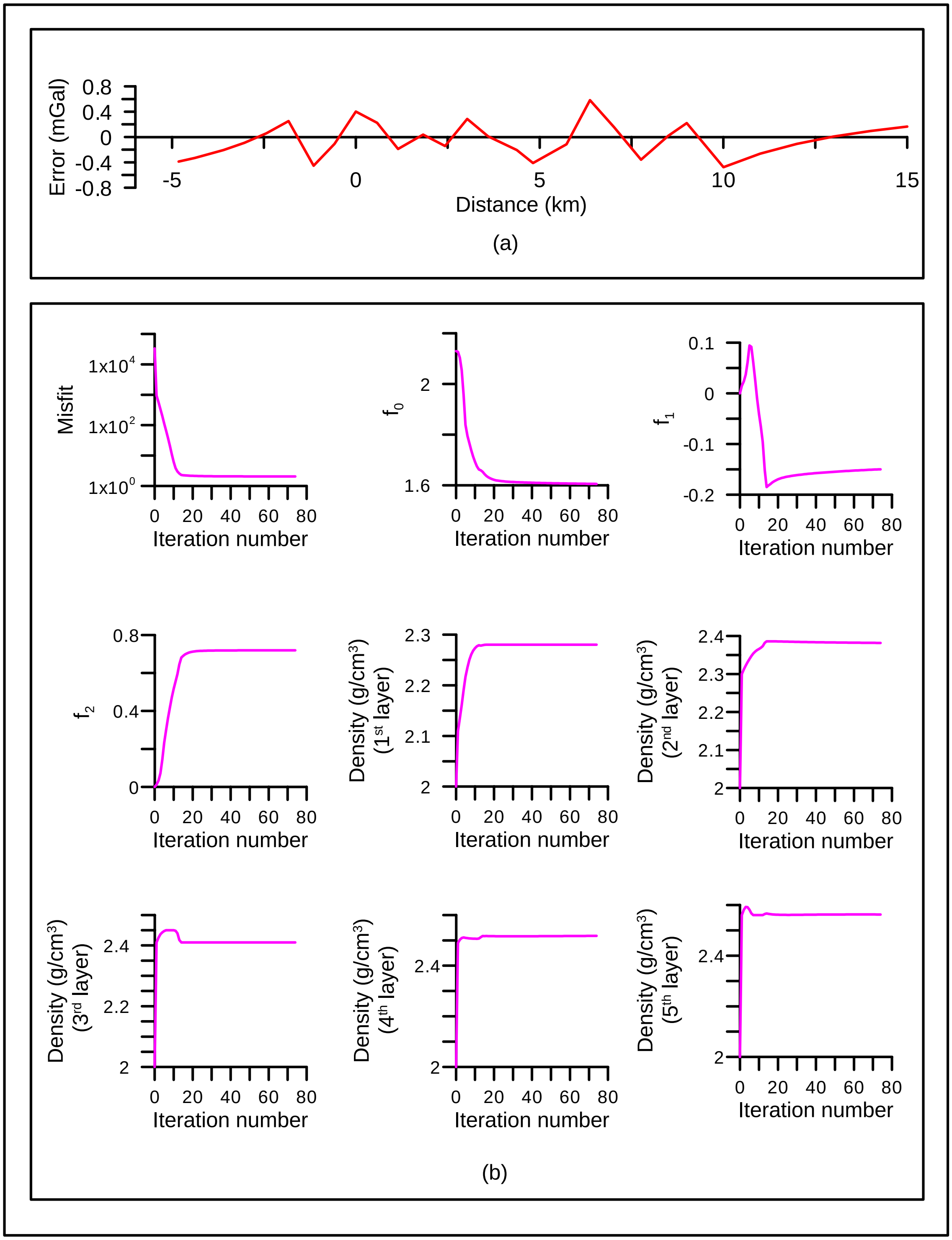


Figure 4.7 (a) Error analysis between the observed and modeled gravity anomalies across the Aswaraopet master fault morphology, Chintalpudi subbasin, India, (b) changes in misfit, coefficients of a 2nd degree polynomial, and densities of subsurface formations against the iteration number.

approximate parameters significantly differ in magnitude from the measured quantities.

Table 4.5
Measured and estimated depths to density interfaces, Chintalpudi subbasin, India

Formation	Measured depth (km)	Initial depth (km)	Estimated depth (km)	Error (%)
1	0.46	0.2	0.43	6.5
2	1.265	0.9	1.10	13.0
3	1.835	1.2	1.87	1.9
4	2.54	2	2.33	8.3
5	2.935	2.5	3.01	2.5

The algorithm had identified the approximate location of the fault plane at 2.13 km in each case. Initially, this value was assigned to the first coefficient of the polynomial in either case, whereas the other coefficients were set to zero. The algorithm had performed 74 and 16 iterations respectively before terminating. The estimated parameters remained more or less unchanged beyond the concluding iterations (Figure 4.7b and Figure 4.9b). The modeled gravity anomalies are shown in Figure 4.6a and Figure 4.8a as solid lines. The fit between the observed and modeled gravity anomalies in either case is satisfactory (Figure 4.6a and Figure 4.8a). A maximum error of 0.58 mGal between the observed and modeled gravity anomalies is observed at 6.3 km on the profile (Figure 4.7a) when the inversion was performed to estimate the densities and fault plane geometry, whereas a maximum error of 0.64 mGal (absolute) is observed at the 10^{th} km (Figure 4.9a) when the anomalies are inverted for depths and fault plane geometry. The estimated density and depth parameters subsequent to respective inversions are given in Table 4.4 and

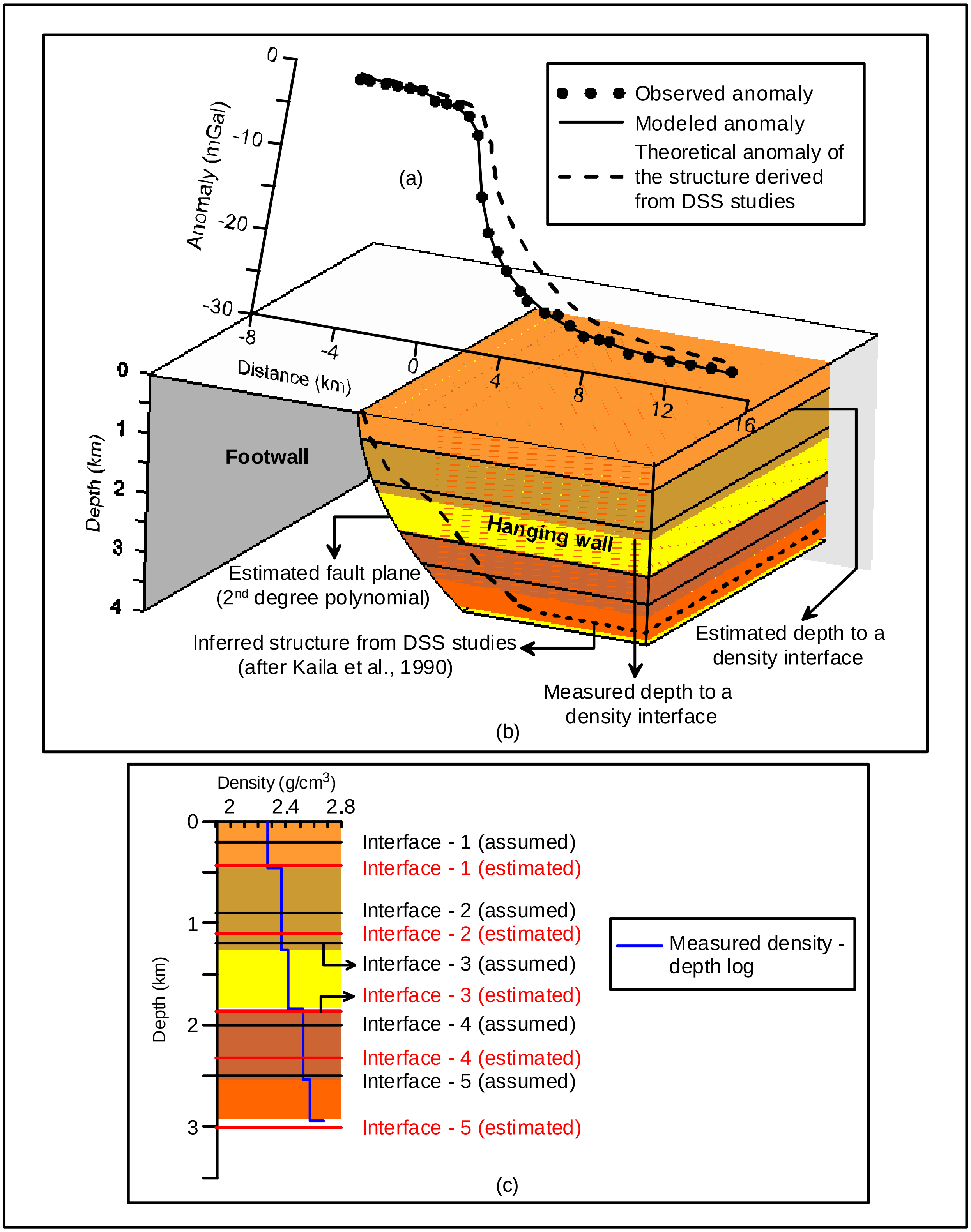


Figure 4.8 (a) Observed and modeled gravity anomalies, (b) inferred geometry of the Aswaraopet master fault, Chintalpudi subbasin, India, (c) density-depth relationship. Anomalies are analyzed to estimate the depths of density interfaces

Table 4.5 and shown in Figure 4.6c and Figure 4.8c respectively. The errors (%) between the estimated and measured parameters in each case are also given in Tables 4.4 and 4.5. The inferred coefficients of the polynomial to construct the fault plane geometry in either case are tabulated in Table 4.6 and illustrated in Figures 4.6b and 4.8b respectively.

Table 4.6
Estimated coefficients of the 2nd degree polynomial, Chintalpudi subbasin, India.

Coefficient	Estimated coefficients of the 2nd degree polynomial in case of densities and fault plane inversion	Estimated coefficients of the 2nd degree polynomial in case of depths and fault plane inversion
f_0	1.606	1.564
f_1	-0.149	0.143
f_2	0.719	0.506

When the anomalies are subjected for inversion to estimate densities and the fault plane geometry, the modeled densities of the first and second formations are slightly overestimated (~0.4%) while others marginally underestimated (Table 4.4 and Figure 4.6c).

On the other hand, when the inversion was performed for estimating both depths and fault plane geometry simultaneously, the modeled depths of the first, second and fourth density interfaces are modestly underestimated whereas the third and fifth density interfaces are slightly overestimated (Table 4.5 and Figure 4.8c). The changes in the estimated parameters with the iteration number in each case are shown in Figures 4.7b and Figure 4.9b respectively.

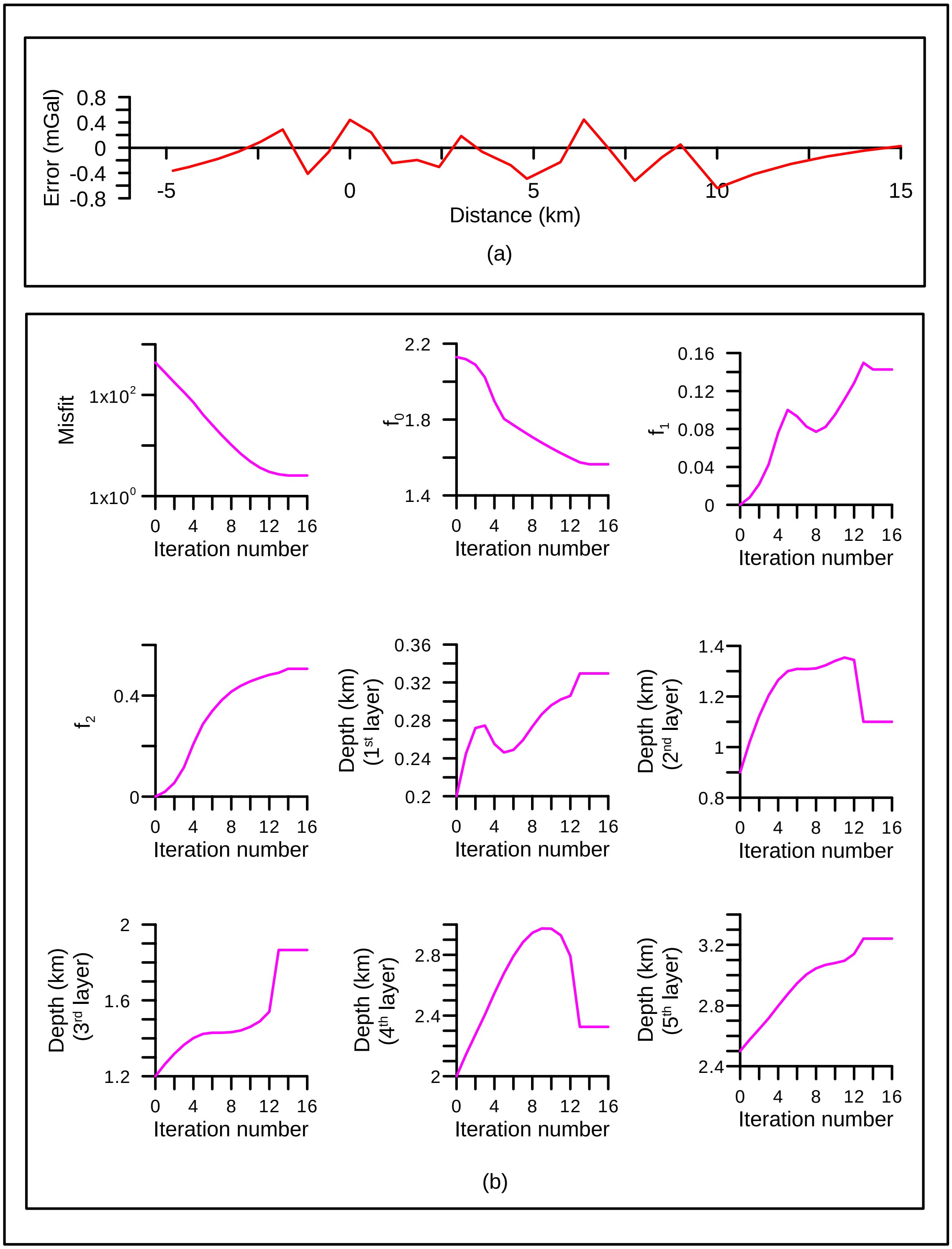


Figure 4.9 (a) Error analysis between the observed and modeled gravity anomalies across the Aswaraopet master fault morphology, Chintalpudi subbasin, India, (b) changes in misfit, coefficients of a 2nd degree polynomial, and depths to various density interfaces against the iteration number.

Furthermore, the inferred structure of the basin across the Aswaraopet fault from DSS studies (after Kaila et al. 1990) is also shown in Figure 4.8b for comparison. The theoretical gravity response of this structure is shown as a dashed line in Figure 4.8a along with the observed anomaly. It can be seen from Figure 4.8a that the modeled gravity anomalies of the structure from present inversion closely mimic the observed ones, whereas the gravity response of the seismically derived structure (Kaila et al., 1990) does not. In addition, the large gradient (4.5 mGal/km) observed in the anomaly across the fault plane does not agree well with the interpreted model of Kaila et al. (1990), whereas it agrees reasonably well with the present inversion result. Further, the error (4.6%) between the measured and estimated thickness of the basin from DSS studies near the existing deep borehole is relatively more than the one estimated (2.55%) from the present inversion.

4.5 Results and Discussion

A gravity inversion technique and related software, GRAVLIS, are developed to simultaneously estimate the geometries of fault planes and the parameters pertaining to either densities or depths of various subsurface formations of strike limited listric fault sources from the observed gravity anomalies. The advantage of the proposed algorithm is that it can be used to analyze the gravity anomalies of the structures even when the profile along which the interpretation is intended fails to bisect the fault plane.

The algorithm is applied to both synthetic and real field gravity anomalies. In case of synthetic example; significant level of pseudorandom noise was added to the gravity anomalies produced by a structure, whose fault plane was described with a 6th degree polynomial. To study the effect of the choice of the degree polynomial in the interpretation, the noisy anomalies were inverted presuming a 2nd degree polynomial for the fault plane. The noisy anomalies were then analyzed to estimate i) the densities and fault plane geometry, keeping the depths of density interfaces unchanged, and ii) depths and fault plane geometry, keeping the densities intact. In either case, the estimated parameters pertaining either to densities or depths closely mimic the assumed parameters, although random noise is present in the anomaly. However, the choice of a lower order polynomial lead to underestimate the amount extension across the normal fault, when inversion is performed to recover the fault plane geometry and depths of density interfaces.

The observed gravity anomalies across the Aswaraopet master fault from the eastern margin of the Chintalpudi subbasin in India are analyzed by the proposed inversion and found that the estimated parameters (densities and thicknesses of subsurface formations within the hanging wall) from independent gravity inversion reasonably coincide with the measured ones. On the other hand, the calculated gravity response of the structure derived from DSS investigations (Kaila et al. 1990) using the measured density-depth data was significantly deviated from the observed anomaly. Further, the large gradient in the observed gravity anomaly over the fault plane is better explained by the present interpreted model rather than the one reported from seismic data interpretation (Kaila et al., 1990).

By and large, the modeled structure of the Aswaraopet fault morphology from gravity inversion closely resembles the one obtained from interactive gravity modeling (Chapter-III). However, given a choice, the estimated model from gravity inversion is more preferable because the model gravity anomalies in case of inversion closely replicate the observed anomalies; whereas it is not so prominent in case of interactive modeling.

CHAPTER

FIVE

Automatic gravity modeling of sedimentary basins by means of polygonal source and exponential density contrast model *

5.1 General

Estimation of sediment thickness and underlying bedrock topography from the observed gravity anomalies are important in hydrocarbon exploration to find out the location of possible stratigraphic traps (Silva et al., 2010), in glaciology to infer the base flow rate of discharge (Krimmel,1970; Stern, 1978; Venteris and Miller, 1993), in landfill analysis as a tool for density determination (Mantlík et al., 2009), in hydrogeological investigations to understand the geological structures of aquifers (Adema et al., 2007; Bohidar et al., 2001), in ground motion amplification studies to study source characterization (Torizin et al. 2009; Jacoby and Smilde, 2009; Aydemir et al., 2014) etc.

In general, sedimentary rocks have densities lower than the basement rocks, in which case, negative gravity anomalies are usually observed over

----

** Journal of Applied Geophysics (Elsevier), DOI: 10.1016/j.jappgeo.2015.11.007*

thick sedimentary basins. These anomalies can be analyzed not only to map the basin boundaries but also to determine the depth distribution of sedimentary basins.

5.2 Status of existing interpretational techniques

Interpretation of gravity anomalies of sedimentary basin can be realized by direct and indirect methods. More often than not the terms modeling and inversion are used more or less synonymously to refer to direct methods of gravity interpretation to trace the boundaries or outlines of anomalous bodies using iterative approach, although they differ from each other in their operation principles (Murthy, 1998; Chakravarthi et al., 2013a).

In direct methods, many algorithms employ either the stacked prism model of Bott (1960) or the polygonal model of Talwani et al. (1959) to analyze the gravity anomalies of sedimentary basins. The techniques developed by Murthy et al. (1988), Pilkington and Crossley (1986), Murthy et al. (1990), Leão et al. (1996), Barbosa et al. (1997, 1999), Annecchione et al. (2001), Mendonca (2004), Gabalda et al. (2005), Silva Dias et al. (2007), Osman et al. (2007) are based on the assumption that the sedimentary load over the basement possess uniform density; hence these techniques fall short of the modeling needs for gravity anomalies in sedimentary basins, where geologic settings warrant the use of variable density. It is also true that gradual increase in density with depth necessitates the introduction of discrete bodies which differ only in having slightly different densities that may not necessarily correspond to geologically distinct units.

In this context, Rao (1986) used a quadratic density function to analyze the gravity anomalies of sedimentary basins, whereas Pan (1989) employed constant horizontal density gradient to compute the gravity anomalies of irregularly shaped 2D bodies, Reamer and Ferguson (1989), Hansen (1999), Hamayun et al. (2009), D'Urso (2014a) adopted linear density-depth relationship, Kwok (1991) considered linear density variation with horizontal position and depth to compute the gravity anomalies using complex contour integrals, Oliva and Ravazzoli (1997) proposed a complex variables formulation for the computation of the gravity effect of 2D bodies having densities varying both laterally and with depth, Zhang et al. (2001) make use of a polynomial function of arbitrary degree expressed in terms of depth and lateral position to calculate the gravity anomalies of 2D polygonal bodies, Zhou (2008) used line integrals to calculate the gravity anomaly of a 2D mass of complicated geometry and spatially variable density contrast, and Zhou (2009) extended the line integral method to compute the gravity anomalies of irregular 2D masses with horizontally and vertically dependent density contrast However, each enlisted method has its own merits and demerits in its application (Chakravarthi, 2009).

García-Abdeslem (2003) had developed a 2D inversion incorporating a cubic density polynomial to analyze the gravity anomalies of geologic sources, where the source-basement geometry was described by the Fourier series. In this case the problem of gravity interpretation reduces to estimate the coefficients of the Fourier series. In recent past, Zhou (2013) has developed an automated iterative forward modeling scheme based on the line integral and maximum difference reduction methods to estimate 2D bedrock topography from a given gravity anomaly profile.

However, due to the fact that gravity anomalies of sedimentary basins cannot be realized in the space domain using EDCM; many existing algorithms perform forward modeling in the frequency domain and then transform the anomalies back to the spatial domain for further analysis. For e.g., Cordell (1973) had developed a recursive method that uses both the gravity field and its vertical derivative (determined by convolution in discrete Fourier series) to solve the structure of a sedimentary basin from the observed gravity anomalies. The method of Chai and Hinze (1988) involved the calculation of gravity anomalies in the wave number domain followed by their transformation to the space domain by a shift-sampling technique. Although, Rao et al. (1993) could derive expressions for the Fourier transforms of the gravity anomalies of a few simple geometric models with EDCM and adopt them to analyze the gravity anomalies of sedimentary basins; these strategies prone to yield unreliable interpretations when sediment-basement interface has major undulations.

On the other hand, a few researchers have accommodated exponential density variation in gravity interpretation by alternative means. For e.g., Murthy and Rao (1979), Rao and Murthy (1989) proposed subdivision of each side of a polygon into a number of segments, among each of which the density contrast was assumed to vary linearly with depth. Guspí (1990) described the exponential density function by a series approximation to compute the gravity response. However, the former technique consumes significant amount of time even for forward modeling (Visweswara Rao et al., 1994); whereas the later one requires the knowledge of the degree of the polynomial, which is not known a *priori*. It also becomes a strenuous task to accommodate exponential density variation in the available commercial software.

Chakravarthi et al. (2013a, 2013c) have extended the Bott's method (stacked prism model) to develop algorithms in the space domain to analyze gravity anomalies of 2.5D and 2D sedimentary basins using EDCM, where forward modeling was performed by a combination of analytical and numerical approaches. However, the Bott's method is strictly applicable only when the anomalies are available at equal station interval; elsewhere, such criteria may or may not be fulfilled. Therefore, a need exists to develop an interpretation strategy in the space domain using EDCM to analyze the gravity anomalies of sedimentary basins, which is free from the enlisted limitations.

In this Chapter, an equation is derived in the space domain to compute the gravity anomalies of 2D sedimentary basins treating the source as a polygonal geometry within which the density contrast varies in a prescribed exponential form. Based on the principles of automatic modeling (Chakravarthi et al., 2015c) an interpretation strategy coupled with relevant software in JAVA is developed to analyze the gravity anomalies of sedimentary basins. The applicability of proposed modeling is demonstrated on both synthetic and real world gravity anomalies. The estimated depth parameters from automatic modeling are compared with the assumed parameters in case of synthetic example and with the available/previously reported information in case of real field example.

5.3 Forward gravity modeling – Theoretical considerations

In Cartesian co-ordinate system let the z-axis is positive vertically downwards and the x-axis transverse to the strike of a 2D sedimentary basin (Figure 5.1). The strike of the basin is along the y-axis perpendicular to the xz plane.

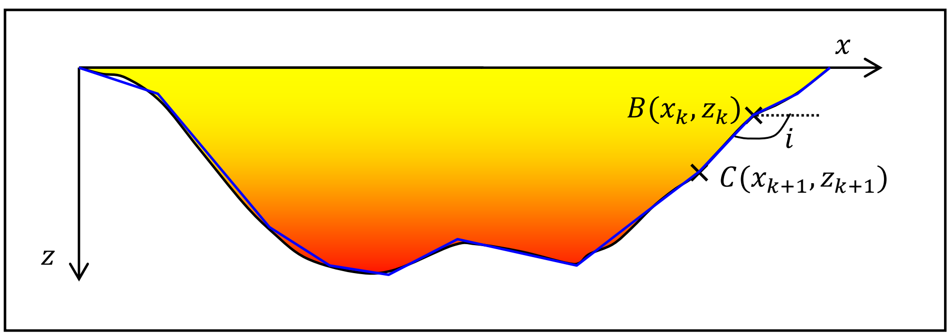


Figure 5.1 Schematic representations of a sedimentary basin (black solid line) and its approximation by a polygon (blue solid line). B and C are the vertices of the Kth side of the polygon. The colour gradation from yellow to red within the structure represents the increase in density with depth

The gravity anomaly of such a sedimentary basin at any point, $P(X_j, Z_j)$, on a profile outside the source region can be obtained by integrating the gravity effect of a 2D element throughout its cross-sectional area as expressed by Chakravarthi et al. (2015c)

$$\Delta g_{2D}(X_j, Z_j) = 2G \int_s \frac{\Delta\rho(z)(z - Z_j)}{(\overline{x - X_j}^2 + \overline{z - Z_j}^2)} dxdz, \tag{5.1}$$

where, G is universal gravitational constant, (x, z) are source coordinates of a 2D element within the source and $dxdz$ represents its cross-sectional area. Here $\Delta\rho(z)$ is the density contrast of the sediments at any depth z represented by equation (1.14).

Upon substitution equation (1.14) for $\Delta\rho(z)$ and applying Stoke's theorem, equation (5.1) becomes

$$\Delta g_{2D}(x_j, z_j) = 2G\Delta\rho_0 \oint_z e^{-\lambda z}\, tan^{-1}\frac{\overline{x - X_J}}{\overline{z - Z_J}}\, dz. \tag{5.2}$$

Approximating the outline of the basin by a polygon BC….(Figure 5.1), the x term in the integrand of equation (5.2) can be expressed for the kth side, such as BC as

$$x = a + z\, coti, \tag{5.3}$$

where, $a = x_k - z_k coti$. Here, i is the angle made by kth side with the x-axis. Furthermore, (x_k, z_k) are the coordinates of the vertex B. The gravity effect of the kth side of the polygon is finally expressed as

$$\Delta g_{BC}(X_j, Z_j) = 2G\Delta\rho_0 \int_{z_k}^{z_{k+1}} e^{-\lambda Z}\, tan^{-1}\frac{\overline{(a + z\, cot\imath) - X_J}}{\overline{z - Z_J}}\, dz. \tag{5.4}$$

Here, z_{k+1} is the depth ordinate of the vertex C, whose coordinate is represented by x_{k+1}. The total gravity effect of the polygon can be obtained as (Chakravarthi et al., 2015c)

$$\Delta g_{2D}(X_j, Z_j) = \sum_{k=1}^{N} \Delta g_k(X_j, Z_j), \tag{5.5}$$

where, N is the number of sides of the polygon. It is to be noted that equation (5.4) needs to be solved numerically because no closed form solution exists for it in the space domain. Further, equation (5.4) and (5.5) are strictly valid for the profile runs transverse to the strike of the basin. In case it runs at an angle, α, with the x-axis, then the anomalous field can be computed by replacing X_j in equation (5.4) by $X_j \cos \alpha$ (Chakravarthi and Ramamma, 2013b). Further, the gravity anomaly of a structure with uniform density can be realized by letting λ=0 in equation (5.4) followed by using equation (5.5). The accuracy of the proposed numerical method of anomaly calculation is demonstrated with a few selected examples in the following sections.

5.3.1 Forward modeling of a homogeneous vertical prism

Figure 5.2a shows the gravity response of a homogeneous vertical prism (Figure 5.2b) at 21 equispaced observations in the interval $X_j \in$ [0 km, 20 km] obtained from both the numerical solution of equation (5.4) and (5.5), and the analytic solution of the prism by Murthy (1998). In this case, the prism is considered to have 5 km width, 4 km depth extent and possess uniform density contrast of -0.45 gm/cm^3. In order to compute the gravity anomalies of the prism by the proposed numerical method, the structure is simulated with a polygon whose vertices are given by P(7.5,0), Q(12.5,0), R(12.5,4) and S(7.5,4). The solid line in red shown in Figure 5.2a represents the anomalies

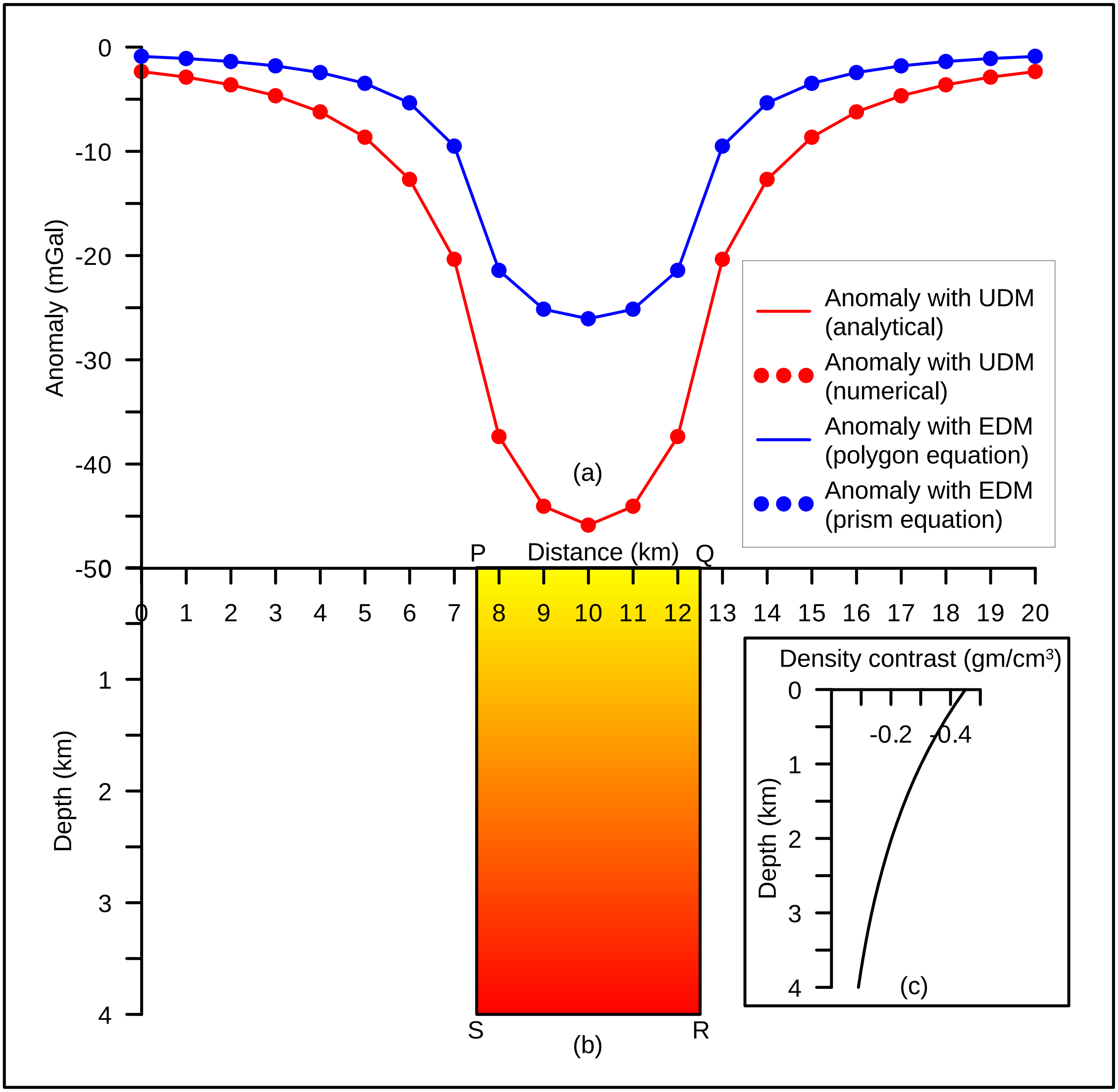


Figure 5.2 (a) Gravity anomalies with uniform and exponential density contrast models, (b) geometry of assumed structure, (c) prescribed EDCM. The color gradation from yellow to red within the prism indicates the increase in density in case of EDCM

obtained from the analytical method and the filled circles in red correspond to the anomalies realized from the numerical method. In this case, the maximum difference between the two anomalies hardly exceeds 5E-05 mGal, which shows the accuracy of the proposed method.

5.3.2 Forward modeling of a vertical prism with EDCM

The gravity anomalies of the same structure (Figure 5.2b) at 21 observations in the interval X_j ∈ [0 km, 20 km] with exponential decay in density contrast obtained from both the present method and the one from numerical solution of the prism (Chakravarthi et al., 2013c) are also shown in Figure 5.2a. In both cases the values of $\Delta\rho_0$ and λ are assumed as -0.45 gm/cm^3 and 0.4 km^{-1} respectively (Figure 5.2c). In this case, the maximum difference between the magnitudes of the two anomalies is 4E-05, which again is insignificant.

5.3.3 Forward modeling of fault and trapezoidal geometries with EDCM

Solid lines in blue in Figures 5.3a and 5.3c represent the gravity anomalies calculated by Rao et al. (1993) over a vertical fault and trapezoidal geometries (Figures 5.3b and 5.3d) using a quadratic density model (QDM). The authors have used a set of quadratic density functions, one defined with -0.4957+ 0.2257z-0.0344z^2 and the other by -0.3654+0.1140z-0.0098z^2 (solid lines in blue in Figures 5.3e and 5.3f) to simulate a prescribed exponential density contrast model (EDCM) -0.5$e^{-0.5z}$ (solid lines in red in Figures 5.3e and 5.3f) over two selected depth ranges 0 km to 2 km, and 2 km to 4 km

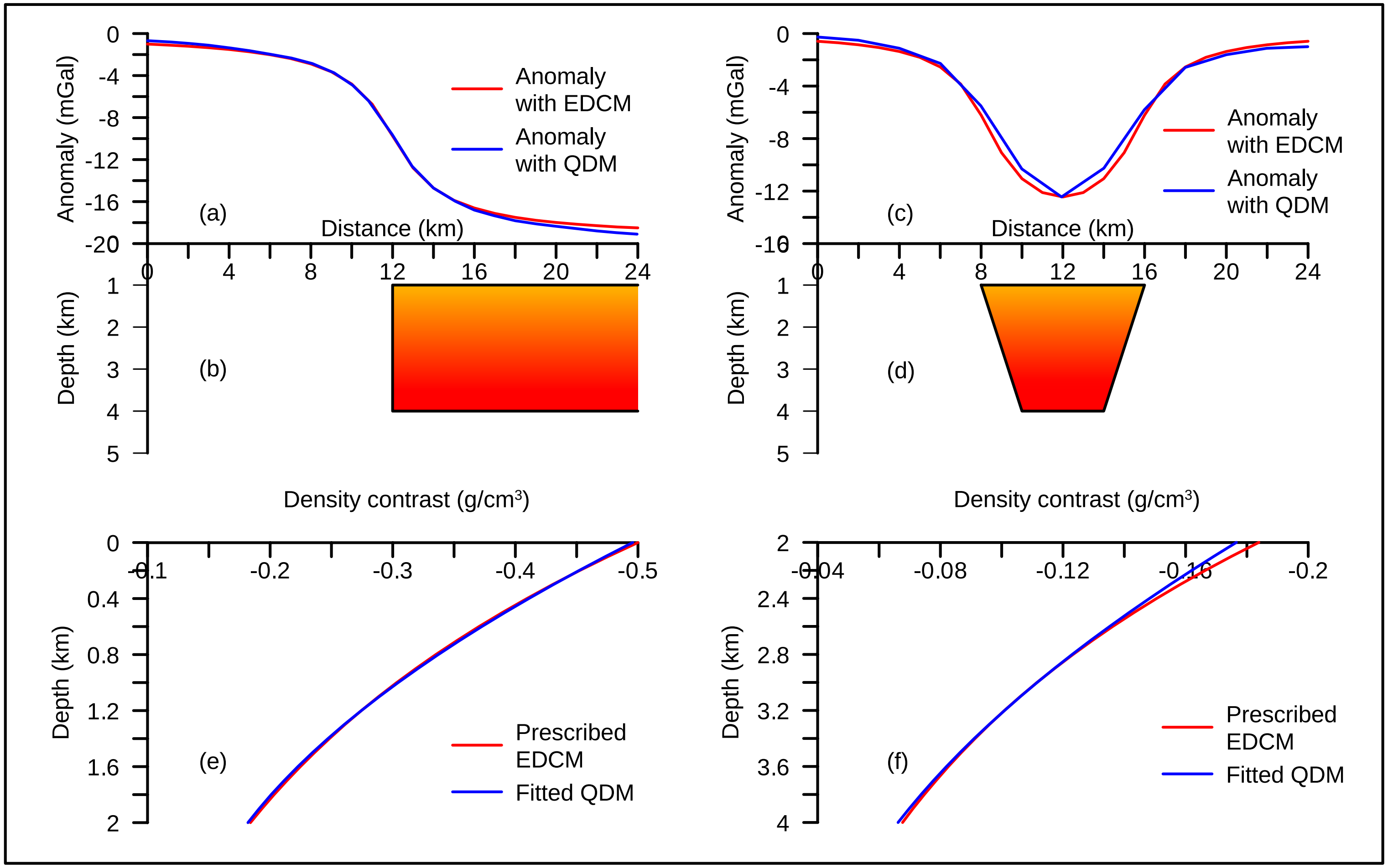


Figure 5.3 Forward gravity modeling based on analytical and numerical approaches (a) and (c) over a vertical fault (b) and trapezoidal (d) models, prescribed exponential density contrast model (EDCM) and fitted quadratic density model (QDM) over two selected depth ranges are shown in (e) and (f). The color gradation from yellow to red within the structures indicates increase in density with depth

respectively. These fitted quadratic density models are then used in respective analytical gravity expressions of the geometries (Rao, 1985; Rao, 1990) to generate the gravity anomalies in the interval $X_j \in$ [0 km, 24 km] in each case as shown in Figures 5.3a and 5.3c. The gravity responses of the two models obtained from the present numerical method using the prescribed EDCM are also shown in Figures 5.3a and 5.3c by solid lines in red.

One can notice from Figure 5.3a and Figure 5.3c that the gravity anomalies obtained from the present method with prescribed EDCM closely coincide with the anomalies obtained with the fitted QDMs by Rao et al. (1993). The marginal deviations noticed between these two anomalies in each case are obvious because the fitted QDM shows moderate deviations from the prescribed EDCM, more particularly so beyond a depth of 2 km (Figures 5.3e and 5.3f). Furthermore, the advantage of the present method is that a single equation is suffice to calculate the gravity anomalies of the two structures (geometries can be simulated with appropriate vertices), whereas the strategies proposed by Rao (1985, 1990) involved the use of different anomaly equations for different geometries.

5.3.4 Forward modeling of a homogeneous sedimentary basin

The filled circles and solid line in red shown in Figure 5.4a correspond to the calculated gravity anomalies over a hypothetical homogeneous sedimentary basin (Figure 5.4b) obtained from the proposed numerical method and the analytical method of Murthy and Rao (1989). In this case, the density

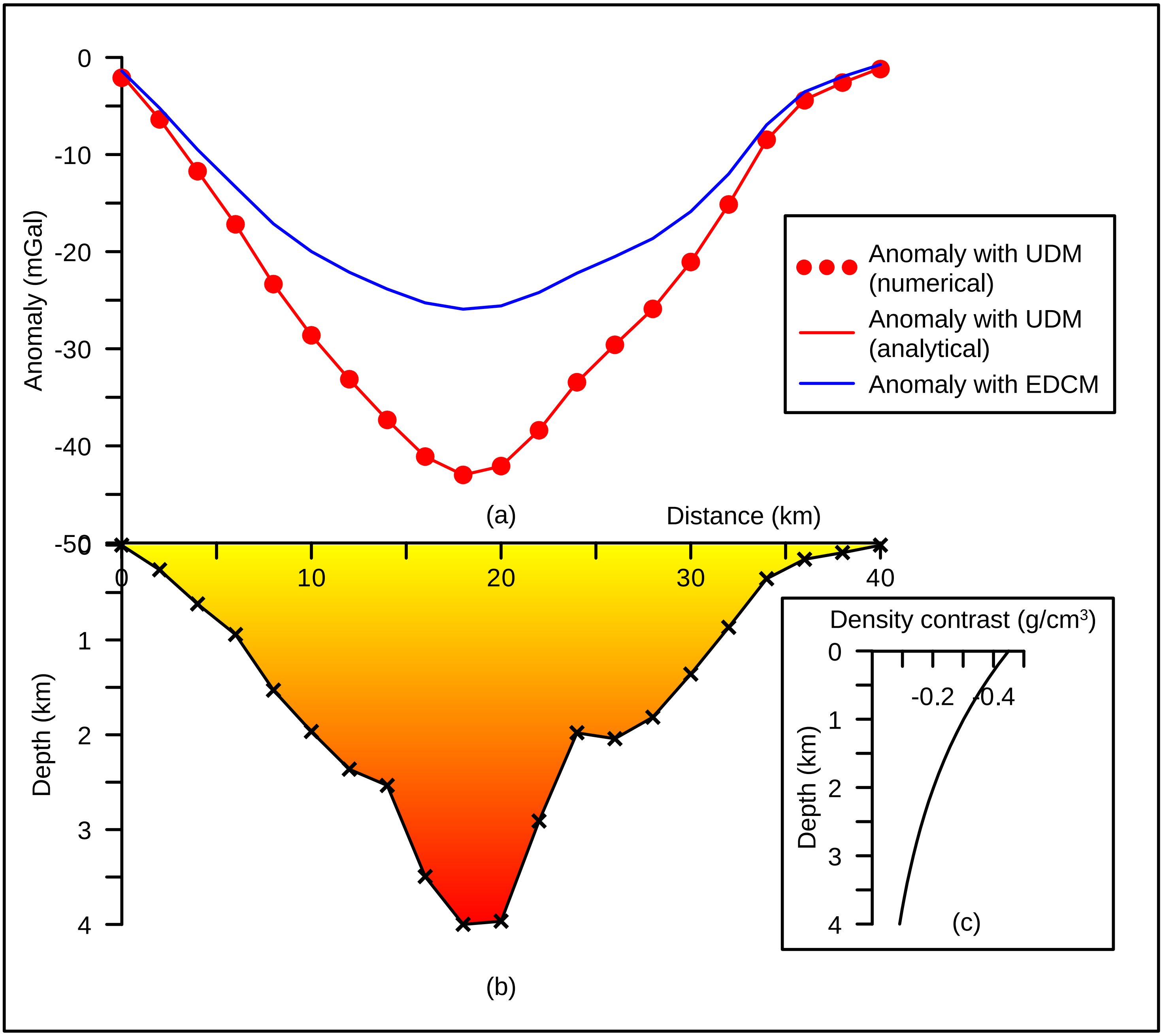


Figure 5.4 (a) Forward gravity modeling with analytical and numerical approaches based on uniform and exponential density contrast models (UDM & EDCM), (b) geometry of a sedimentary basin, (c) prescribed EDCM. The color gradation from yellow to red within the structure indicates increase in density with depth in case of EDCM.

interface (floor of the sedimentary basin) is described with a polygon having 21 vertices as illustrated in Figure 5.4b. In each case, the anomalies are calculated on a plain topography, Z_j= 0, at 21 observations in the interval X_j ∈ [0 km, 40 km] at 2 km spacing using an uniform density contrast of -0.35 gm/cm^3 (Figure 5.4a). One can notice from Figure 5.4a that the anomalies obtained from the present method excellently coincide with those realized from the analytical method of Murthy and Rao (1989). The solid line in blue in Figure 5.4a represents the gravity anomalies produced by the same structure but with exponential decay in density contrast defined with the constants $\Delta\rho_0$ = -0.35 gm/cm^3 and λ =0.35 km^{-1}. The maximum anomaly (absolute magnitude) produced by the basin with uniform density model (UDM) is 43 mGal, whereas it hardly exceeds 26 mGal in case of exponential density contrast model (EDCM) (Figure 5.4a). Thus, sedimentary basins in which the density contrast varies exponentially with depth would generate only moderate gravity anomalies; and hence the interpretation algorithms that are dependent on the magnitude of the anomalous field must consider the exponential density contrast model as a crucial parameter in the analysis for reliable results.

5.4 Automatic modeling of gravity anomalies

The aim of gravity modeling of a sedimentary basin is to estimate the depths to the floor of the basin from the observed gravity anomalies at plurality of observations. The proposed algorithm initializes a sedimentary basin from the observed gravity anomalies and subsequently improves the structure based on the differences between the observed and modeled gravity anomalies within the specified convergence criteria. In the present case, the number of vertices of the polygon equals the number of observations on the profile and distance to each observation on the profile becomes the x coordinate of the corresponding vertex of the polygon. Hence, the problem of gravity modeling simplifies to estimate the optimum depth co-ordinates of the vertices of the polygon.

The initial/approximate depths to the floor of a sedimentary basin are calculated at all observations on the profile by substituting the observed residual gravity anomaly, $g_{obs}(X_{j,}Z_j)$, for, g_{obsmx} in equation (2.4). Having calculated the initial depths, the modeled gravity anomalies, $\Delta g_{2D}(X_{j,}Z_j)$, of a sedimentary basin can be realized through equation (5.4) and (5.5). The difference between the observed and modeled gravity anomalies at any observation can be quantified using equation (2.5).

The depths of a sedimentary basin are then automatically improved by using the principles of automatic modeling (Chakravarthi et al., 2013a; 2015c). Accordingly, the basin depth at any observation, $(X_{j,}Z_j)$, can be improved based on the expression

$$Z_B\left(X_{j,}Z_j\right)^{k+1} = Z_B\left(X_{j,}Z_j\right)^k + \frac{ERR\left(X_{j,}Z_j\right)}{2\pi G\Delta\rho\left(Z_B(X_{j,}Z_j)^k\right)}, \tag{5.6}$$

where, k stands for the number of iterations. Here,

$$ERR\left(X_{j,}Z_j\right) = g_{obs}\left(X_{j,}Z_j\right) - \Delta g_{2D}\left(X_{j,}Z_j\right), \tag{5.7}$$

The interpretation process continues until i) the number of iterations completed, or ii) the value of r.m.s. error falls below a predefined allowable error, or iii) the current/existing r.m.s. error exceeds the preceding value.

5.5 Description of software – MOD2DGREXP

The modeling procedure described in section 5.4 is used to develop a GUI based software, MOD2DGREXP, coded in JAVA to interpret the gravity anomalies of 2D sedimentary basins using EDCM (Annexure 5-A). The architecture of the software is based on the MVC (Model-View-Controller) pattern shown in Figure 2.2. The module ‘Model’ calculates the initial/ approximate depths, computes the gravity response of the structure, and executes the business logic of the algorithm to improve the model space. The role of ‘View’ module is to read the input parameters and display the output in both graphical and ASCII formats. The ‘Controller’ passes the required actions to the view and model modules.

The view module of the code appears on the monitor as shown in Figure 5.5 after invoking the batch file. The input layout of the view module consists in ten input fields namely the area name, profile name, number of observations, distance to each observation expressed in km, observed gravity anomalies in mGal, surface density contrast in gm/cm^3, Lambda in km^{-1}, minimum and maximum permissible depths expressed in km and the number of iterations to be performed. The graphical layout is arranged into anomaly, structural, misfit and density-depth panels respectively as depicted in Figure 5.5. The ASCII form of the output shall be displayed in the ASCII layout.

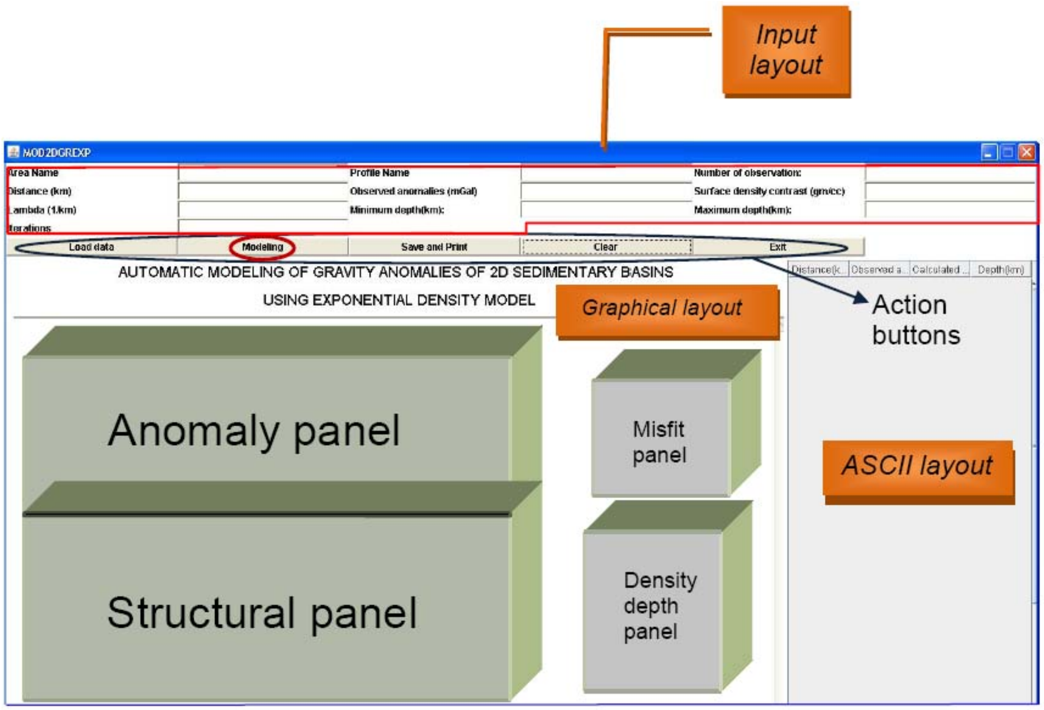


Figure 5.5 View module of MOD2DGREXP

The user enters the input parameters either by using the respective fields of the input layout or specifies the parameters in a formatted Excel sheet and reads the file to the code by means of 'Load data' action button. The interpreter invokes the action button 'Modeling' to perform the interpretation.

The growth in the model space shall be displayed in an animated form along with the improvements in modeled anomalies against the iteration number. The changes in misfit and density contrast shall also be displayed in animated forms in respective panels. The user saves the output in both html and jpg formats and opts for printing by invoking 'Save and print' action button.

5.6 Applications

The applicability of proposed method of automatic modeling is demonstrated on both synthetic and real world gravity anomalies. The real field gravity data pertains to the San Jacinto graben, California (after Cordell, 1973).

5.6.1 Synthetic example

Figure 5.6a shows, in the interval X_j ∈ [0 km, 60 km], 25 randomly distributed noisy gravity anomalies (solid circles) produced by a synthetic model of sedimentary basin, whose geometry is shown in Figure 5.6b. The assumed structure represents a typical rifted sedimentary basin, whose basement consists in numerous discontinuities caused by several synthetic and antithetic faults. Furthermore, these faults observed in the basement in the interval X_j ∈ [15 km, 42 km] have minor throws and show relatively complex pattern compared to the marginal faults of the basin. In this case, pseudorandom noise was Gaussian with zero mean and a standard deviation of 0.11 mGal. The density contrast within structure obeys exponential decrease with depth following equation (1.14) defined by the constants $\Delta\rho_0$= -0.45 gm/cm^3 and λ= 0.4 km^{-1}, respectively (Figure 5.6c).

The noisy anomalies shown in Figure 5.6a are analyzed by the present method to recover the basement structure. The constants of the prescribed EDCM remain unchanged during the analysis, whereas the depths of the basin are allowed to vary within the specified bounds as specified in section 5.5. The modeling algorithm has performed 14 iterations before it got terminated. The

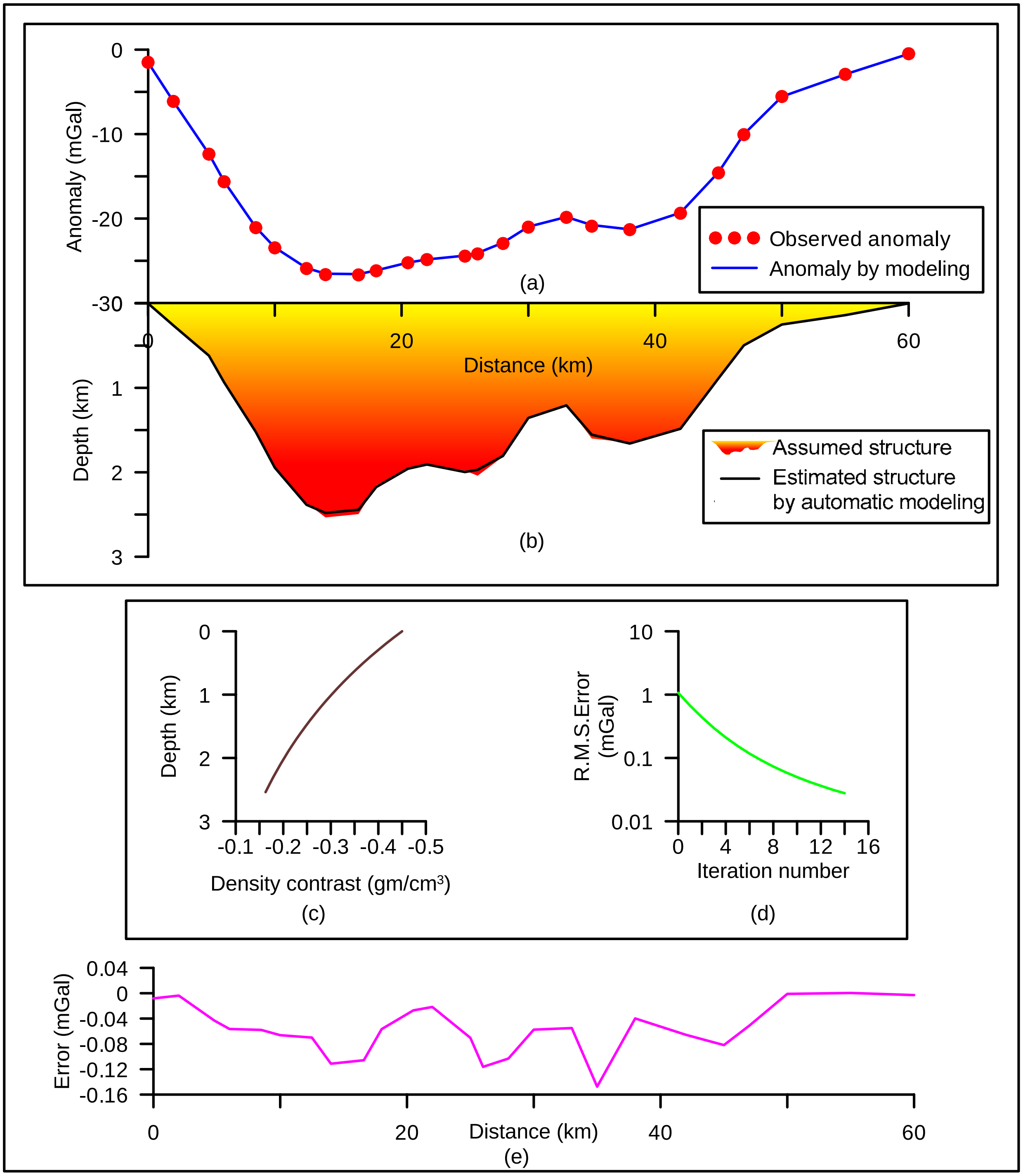


Figure 5.6 (a) Observed and modeled noisy gravity anomalies by automatic modeling, (b) assumed and estimated structures, (c) prescribed EDCM, (d) changes in misfit with iteration, (e) error between observed and modeled anomalies at the end of 14th iteration. The color gradation from yellow to red within the structure indicates increase in density

misfit (equation 2.5) had reduced from 1.08 mGal to 0.02 mGal at the end of the 14th iteration (Figure 5.6d), beyond which the new misfit got exceeded its previous value thereby the modeling process ended. The model gravity anomalies and the observed anomalies show good correlation with each other at the end of the concluding iteration (Figure 5.6a), with a maximum error (absolute) between them (~0.15 mGal) observed at the 35th km. By and large, the algorithm had recovered the structure successfully over the length of the profile, however, with a few exceptions. For e.g. the inferred structure had shown low to moderate deviations from the assumed one in and around the depocentre of the basin in the interval X_j Є [12 km, 16 km]. Such deviations are tolerable considering the presence of significant level of random noise in the observed anomaly.

A sample output generated by the code in html format is shown in Annexure 5-B.

5.6.2 Field Example

The solid dots shown in the interval X_j Є [0 km, 9.982 km] in Figure 5.7a represent the observed gravity anomalies of the San Jacinto graben, California (after Cordell, 1973). The density contrast-depth data of the graben derived from seismic refraction surveys by Fett (Unpublished thesis, University of California at Riverside, 1968) as reported by Cordell (1973) was shown as a step line in green in Figure 5.7c. Cordell (1973) had used two exponential density contrast models (EDCMs); one defined with $\Delta\rho_0$ = -0.55 gm/cm^3 and

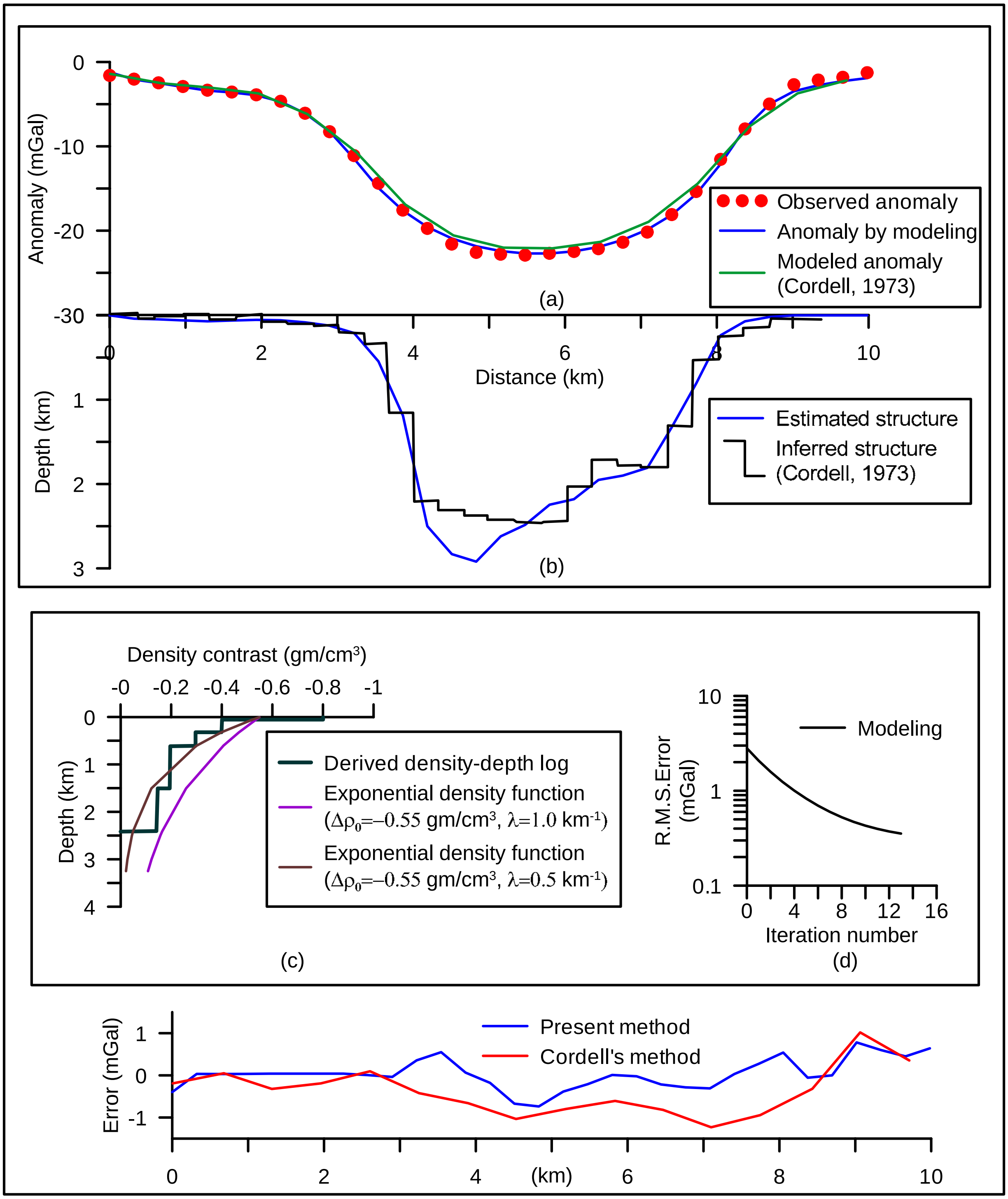


Figure 5.7(a) Observed and theoretical gravity anomalies by automatic modeling, San Jacinto graben, California. Modeled anomalies by Cordell (1973) are also shown for comparison, (b) estimated structures by the present method and Cordell's (1973), (c) derived density contrast-depth data and fitted EDCMs, (d) changes in r.m.s. error with iteration, (e) comparison of errors in anomaly observed in case of the present and Cordell's (1973) methods

$\lambda = 0.5\ km^{-1}$ (shown as a solid line in purple in Figure 5.7c) and the other with $\Delta\rho_0 = -0.55\ gm/cm^3$ and $\lambda = 1.0\ km^{-1}$ (shown as a solid line in brown in Figure 5.7c) to describe the derived density contrast-depth data of the graben. He had shown that the use of former density model in the gravity interpretation had yielded a structural solution (shown as a step line in Figure 5.7b) that was consistent with seismically derived information.

In the present case, we have analyzed the observed gravity data using the proposed automatic modeling to decipher the basement configuration of the graben with an aim to examine whether the inferred model is consistent with the one reported by Cordell (1973). The prescribed EDCM defined with $\Delta\rho_0 = -0.55\ gm/cm^3$ and $\lambda = 0.5\ km^{-1}$ was used in the present analysis to describe the density contrast variation with depth. For such an interpretation, the algorithm had performed 13 iterations. Figure 5.7d shows the changes in r.m.s. error with the iteration number. The modeled gravity anomalies at the end of the concluding iteration are shown in Figure 5.7a and the estimated basement configuration of the graben in Figure 5.7b respectively. The theoretical gravity anomalies and corresponding inferred structure realized by Cordell (1973) were also shown in Figures 5.7a and 5.7b for comparison. The maximum depth to the basement estimated from the present method is 2.9 km against a figure of 2.4 km reported by Cordell (1973).

5.7 Results and Discussion

A space domain based algorithm using the principles of automatic modeling is developed along with a GUI based software to analyze the gravity anomalies of sedimentary basins among which the density contrast obeys exponential decrease with depth. The cross-section of a sedimentary basin is viewed as a polygon, whose gravity anomaly equation is derived in the space domain using a combination of analytical and numerical approaches with exponential density contrast model. The proposed algorithm is fully automatic in the sense that it generates the initial structure of a sedimentary basin from the observed gravity anomalies at plurality of observations and improves the structure in an iterative approach based on the differences between the observed and modeled gravity anomalies until the modeled anomalies closely mimic the observed ones. A software named, MOD2DGREXP, coded in JAVA with inbuilt GUI is also developed, which is platform independent. Besides being user friendly the software is easy to operate.

To demonstrate the applicability of the algorithm the gravity anomalies attributed to a synthetic model are analyzed in the presence of pseudorandom noise. It was found from the analysis that, by and large, the algorithm had recovered the structure successfully including the minor faults exist in the basement.

In case of field example, the derived structure of the San Jacinto graben, California from the proposed method was compared with the previously reported model by Cordell (1973). The deciphered structural models from the present analysis and the one reported by Cordell (1973) showed more or less similar morphological features. For e.g., these two models had revealed that the graben is bounded by steeply dipping fault system towards the west and by moderately dipping fault system towards the east. However, unlike the case with the Cordell's (1973) interpretation the present model shows progressive deepening of the basement towards the west (Figure 5.7b). Furthermore, the basement high which was reflected prominently in the Cordell's (1973) model in the interval $X_j \in [6\text{ km}, 7\text{ km}]$ was not repeated in the present inferred model. It is pertinent to mention here that the modeled gravity anomalies from the present analysis almost coincide with the observed anomalies; where as it is not so in case of Cordell's (1973) interpretation (Figure 5.7a). One can also notice from Figure 5.7e that the magnitude of error between the observed and modeled gravity anomalies is more significant in case of Cordell's (1973) interpretation when compared to the present one. The above factors suggest that the basement configuration derived from the present analysis appears be more reliable than the structural solution proposed by Cordell (1973).

Thus the validity and applicability of the proposed technique is justified.

CHAPTER

SIX

Automatic gravity inversion of sedimentary basins by means of growing polygonal source and exponential density contrast model *

6.1 General

Deciphering concealed basement structures under sedimentary load from the measured gravity anomalies is one of the classic applications of the gravity method. Automatic modeling schemes, such as the one described in Chapter-V, are preferred to analyze the gravity anomalies of open-ended bodies only; whereas, the inversion strategies are used to analyze the anomalies of both open ended and closed bodies (Murthy, 1998; Chakravarthi et al., 2013a). Further, in case of automatic modeling the error between the observed and modeled gravity anomalies at any observation is used to improve only one model parameter (such as depth to the density interface) at respective station, while in inversion the errors at all observations are being used to calculate the improvements in all model parameters. Automatic modeling schemes become handicapped in case the

---

* *Journal of Applied Geophysics (Elsevier), DOI: 10.1016/j.jappgeo.2015.11.007*

residual gravity anomaly of an anomalous source is associated with regional gravity component. On the other hand, inversion algorithms can be designed to simultaneously estimate both the parameters defining the anomalous source and regional gravity component (Chakravarthi et al., 2015c).

6.2 Status of existing interpretational techniques

Although, forward modeling schemes to compute the gravity anomalies of a 2D object with uniform density are available (see for eg., Okabe, 1979; Won and Bavis, 1987; Singh, 2002; D'Urso, 2014b), the practical utility of these schemes to analyze the gravity anomalies of sedimentary basins is rather limited because 1) the density of sedimentary rocks varies with depth as detailed in Chapter-I, and 2) the depth to the floor of a sedimentary basin is not known a *priori*. Bhaskara Rao and Venkateswarulu (1974) have proposed a method of gravity interpretation over sedimentary basins; where the gravity anomalies are interpreted treating the basin as a pair of outcropping faults, one each on either margin of the basin dipping inwards. Such a methodology should be adopted with caution because the gravity anomaly of each fault shall influence of the gravity response of the other.

Murthy and Rao (1989) have designed an inversion algorithm to trace the topography of a 2D density interface from the observed gravity anomalies, where the structure was represented by a stack of vertical prisms over a specified mean depth. Based on the inverse theory of Backus and Gilbert (1967, 1968, 1970), Mickus and Peeples (1992) have developed a technique to analyze the gravity data for the lower surface of a sedimentary basin. Inoue et al. (2012) have proposed an inversion strategy for gravity anomalies of a 2D polygonal source with application to sedimentary basins using simulated annealing. In recent past, Rao (2013) has developed a technique based on the Marquardt's (1963) algorithm to process the anomalies of 2D bodies that are bounded by planar surfaces, where the vertical and horizontal components of gravitational attractions were worked out from the gravitational components in the directions perpendicular and parallel to each surface. Again all the above-mentioned techniques presume uniform density for the sedimentary rocks, which is seldom valid in reality. The technique of Grandis and Dahrin (2014) incorporates the 2D algorithm of Talwani et al. (1959) for forward modeling and the axes of the anomalous mass concentration as constraints in inversion.

In this Chapter, a new interpretation technique is developed based on the principles of inversion (described in section 2.4 of Chapter-II) to analyze the gravity anomalies of sedimentary basins using a prescribed exponential density contrast model (EDCM) followed by the development of relevant software. The proposed approach was then applied to recover the structure from the gravity data produced by a simulated graben defined by steeply dipping boundary faults. The method's ability was also demonstrated on a gravity profile across the San Jacinto graben in California.

6.3 Inversion

The process of inversion of gravity anomalies of sedimentary basins is analogous to a mathematical procedure of finding the depths of a sedimentary basin from the observed gravity anomalies using a prescribed exponential density contrast model (Chakravarthi et al., 2015c).

The inversion process begins with the initialization of a sedimentary basin (based on the Bouguer slab approximation) followed by the calculation of its gravity response as described in section 5.4 of Chapter-V. Because the initial depths are only ballpark, the model gravity anomalies realized through equation (5.4) and (5.5) clearly depart from the observed anomalies. The difference between the two anomalies at any observation, $(X_{j,}Z_j)$, can be expressed as a sum of the products of vertical gradient of anomaly and improvements in depth coordinates of the vertices of the polygon (Chakravarthi et al., 2015c) as

$$g_{obs}(X_j,Z_j) - \Delta g_{2D}(X_j,Z_j) = \sum_{k=1}^{N} \frac{\partial \Delta g_{2D}(X_j,Z_j)}{\partial z_k} dz_k. \qquad (6.1)$$

Linear equation similar to (6.1) is constructed for each observation and N normal equations are framed and solved for the improvements in N depth ordinates of the polygon by minimizing the r.m.s. error defined by equation (2.5) using the principles of inversion as described in Chapter-II. The system of normal equations is finally expressed as (Chakravarthi et al., 2015c)

$$\sum_{j}^{N}\sum_{m=1}^{N}\frac{\partial\Delta g_{2D}(X_j,Z_j)}{\partial a_{j'}}\frac{\partial\Delta g_{2D}(X_j,Z_j)}{\partial a_m}\ (1+\tau_{mj}\delta)da_m$$

$$=\sum_{j=1}^{N}Err(X_j,Z_j)\frac{\partial\Delta g_{2D}(X_j,Z_j)}{\partial a_{j'}}, j'=1,2,\ldots,N. \tag{6.2}$$

The partial derivatives required in equation (6.2) are evaluated numerically, which involves the calculation of the rate of change of the gravity anomaly with respect to each ordinate of the vertex of the polygon as described by Chakravarthi et al. (2015c). The improvements in depth ordinates of the polygon, da_m $(= dz_m)$, $m=$ 1, 2,…, N obtained from the solution of equation (6.2) are used to update the model space in an iterative approach until one of the termination criteria of the inversion is fulfilled as described in Chapter-II.

6.4 Description of software – IN2DGREXP

Based on the proposed methodology of inversion enumerated in section 6.3, a GUI based software, IN2DGREXP, coded in JAVA is developed to analyze the gravity anomalies produced by a sedimentary basin using a prescribed EDCM (Annexure 6-A). The software is fully automatic as it generates the initial structure and improves it iteratively within the specified convergence criteria without any manual intervention. The software works on the MVC pattern as shown in Figure 2.2. The module 'Model' performs the task of finding the initial depths of a sedimentary basin and computes the gravity response, frames and solves the normal equations for the improvements in the depth ordinates, and updates the depth parameters and model anomalies. The roles of ‘View’ and ‘Controller’ modules are described in section 5.5 of Chapter-V.

Once the batch file of the software is call upon, the view module appears on the monitor of the computing machine as shown below

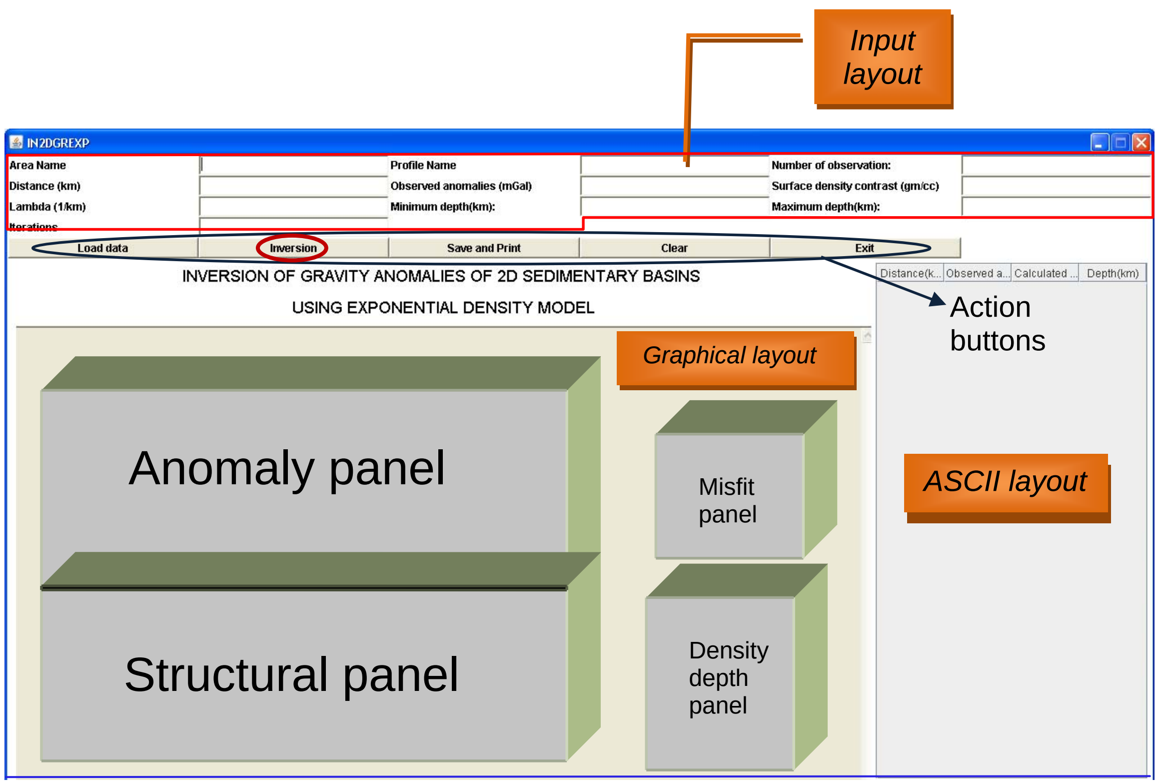


Figure 6.1 View module of IN2DGREXP

As in the case of MOD2DGREXP, the user may enter the input parameters to the code in two ways, either by using the fields of the input layout or by means of a formatted Microsoft Excel sheet. In case the input is stored in an Excel sheet, the same shall be read into the code by invoking ‘Load data’ action button (Figure 6.1). The details of the input parameters are described in section 5.5 of Chapter-V.

Upon activation, the action button ‘Inversion’ (Figure 6.1) performs the task of analyzing the gravity anomalies for the basement configuration of a sedimentary basin using the principles of inversion. The anomaly and structural panels of the graphical layout display the animated versions of the model improvement and corresponding changes in model anomaly. The changes in EDCM and the misfit shall also be displayed in animated forms in respective panels during the process of inversion.

6.5 Applications

Interpretation of two gravity anomalies, one over a synthetic geometry of a sedimentary basin and the other over a real-world case, demonstrates the practical applicability of the proposed inversion.

6.5.1 Synthetic example

The efficacy of the inversion technique and the software are exemplified with the noisy gravity anomalies, which were analyzed previously in section 5.6.1 by automatic modeling. The objective of choosing the same anomaly is to ascertain whether or not the estimated structure from the inversion is comparable to the one obtained from automatic modeling and also with the assumed structure. In this case, the inversion took only 7 iterations for proper a convergence of theoretical anomalies with the observed ones against 14 iterations in case of automatic modeling. The r.m.s. error, which attained a value of 1.08 mGal for the initial structure was reduced drastically to a value less than the predefined allowable error at the end of the 7th iteration (Figure 6.2c), hence, the algorithm got terminated.

The gravity response of the estimated structure after the inversion is shown in Figure 6.2a along with the observed noisy gravity anomaly. The fit between these two anomalies is agreeable. The inferred structure of the basin for which a minimum misfit obtained was shown in Figure 6.2b together with the assumed structure. It was noticed that the estimated depths of the basin and the fit between the observed and modeled gravity anomalies do not exhibit appreciable changes subsequent to 7th iteration, being the concluding one. The residuals, defined as the observed minus the modeled gravity anomalies, after the inversion are shown graphically in Figure 6.2d. A maximum error (absolute magnitude) of 0.04 mGal between the two anomalies is observed at the 16th km on the profile. Minor deviations in the estimated depths from the true ones particularly in the interval $X_j \in [12\text{ km}, 16\text{ km}]$ and at the 25th km are tolerable considering the fact that the anomalies used in the inversion are noisy.

The estimated structure from inversion (Figure 6.2b) remarkably coincides with the structure predicted from automatic modeling (Figure 5.6b) and also compares sensibly well with the assumed structure. In short, the gravity anomaly of the synthetic model analyzed both by automatic modeling and inversion yields exactly the same results.

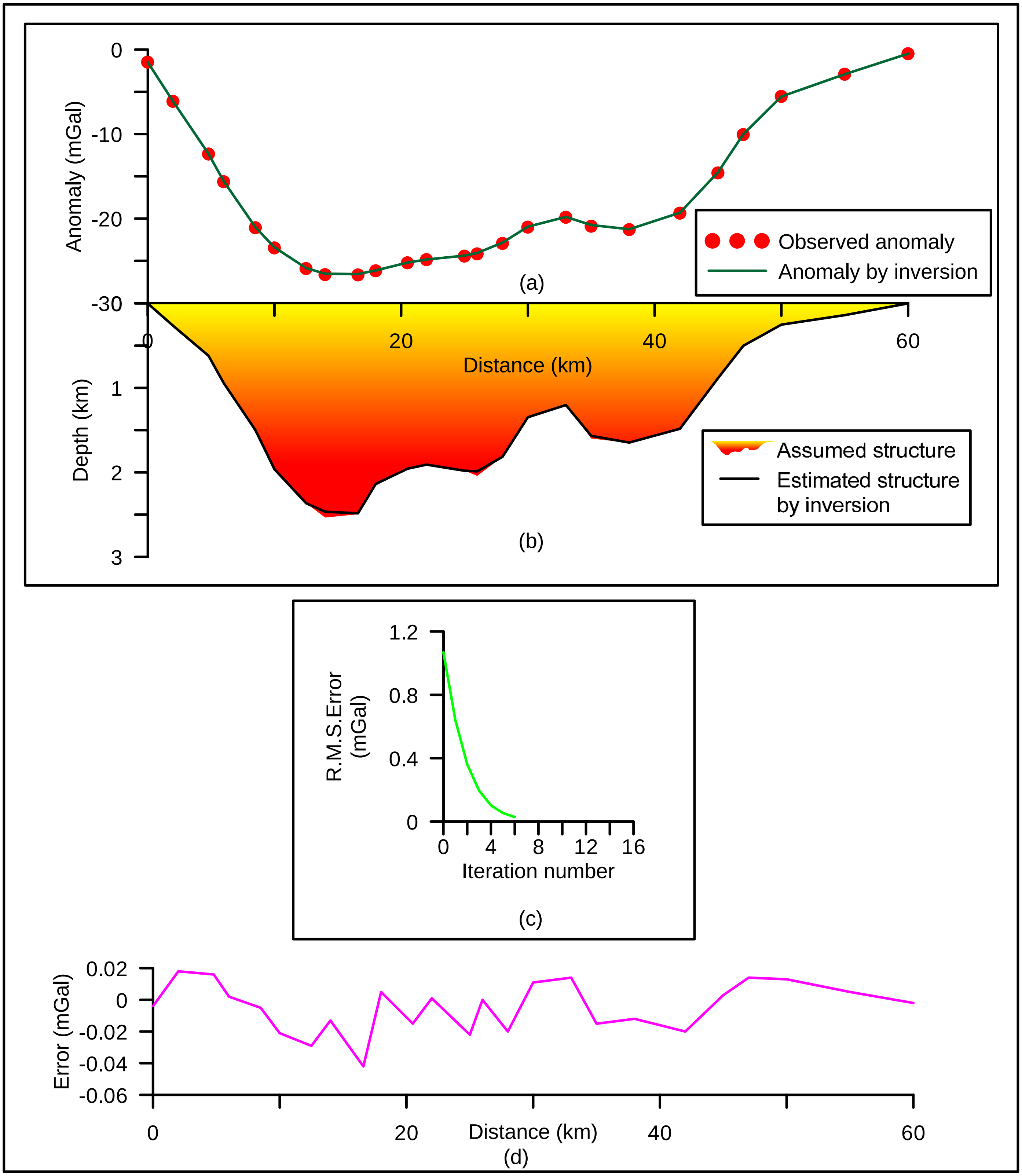


Figure 6.2 (a) Observed and theoretical noisy gravity anomalies by inversion, (b) assumed and estimated structures, (c) changes in misfit with iteration, (d) error between the observed and modeled anomalies at the end of 7th iteration. The description for color gradation from yellow to red within the structure is given in Figure 5.6.

6.5.2 Field Example

The inversion technique is applied to analyze the real field gravity data measured over the San Jacinto graben in California (Figure 5.7a) using a predefined EDCM and the interpretation is shown in Figure 6.2. It is to note that the same anomaly was interpreted in section 5.6.2 of Chapter-V by automatic modeling.

The gravity anomaly of the graben (Cordell, 1973) shown in Figure 6.3a when subjected to inversion with a prescribed EDCM (Figure 5.7c) the algorithm had performed 7 iterations. It was observed that the estimated structure did not show noticeable changes beyond 7th iteration. The modeled gravity anomaly of the structure subsequent to the inversion is shown in Figure 6.3a along with the observed anomaly and the inverted depth structure of the graben in Figure 6.3b, respectively. The inferred structure of the graben and corresponding gravity anomaly by Cordell (1973) were also shown in Figure 6.3a and 6.3b. One can notice from Figure 6.3a that the gravity response of the estimated structure from the present inversion closely resembles the observed gravity anomaly in comparison to Cordell's (1973) interpretation. The r.m.s. error between the observed and calculated gravity anomalies for the starting model was 2.81 mGal (Figure 6.3c) as observed in the case of automatic modeling. However, the decay of r.m.s. error with iteration was found to be rapid in case of inversion compared to automatic modeling (Figures 5.7d and 6.3c). Figure 6.3d compares the

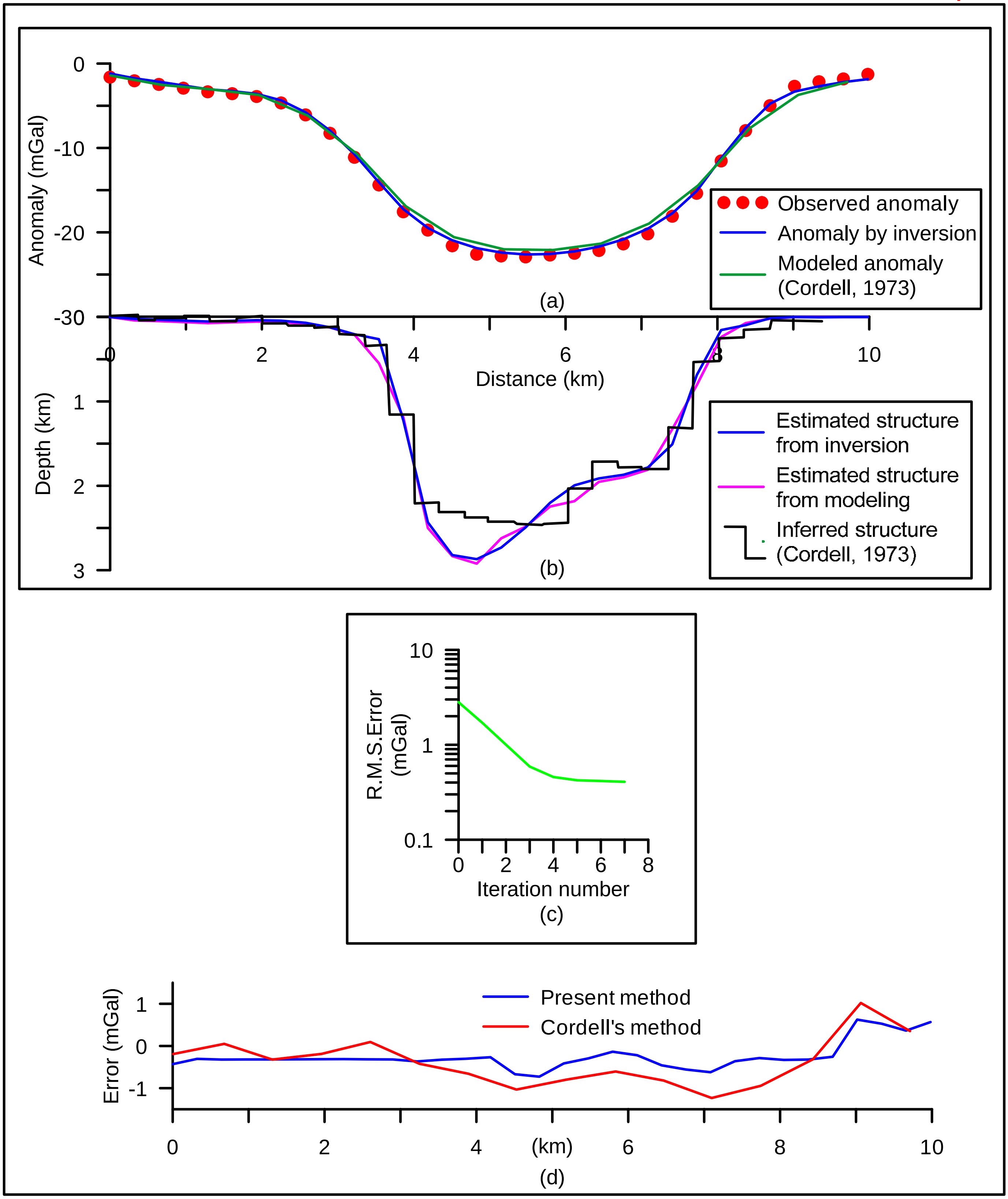


Figure 6.3 (a) Observed and theoretical gravity anomalies by inversion, San Jacinto graben, California. Modeled anomalies by Cordell (1973) are also shown for comparison, (b) estimated structure by present inversion. Depth structures inferred by Cordell (1973), and by automatic modeling (Chapter-V) are shown, (c) changes in r.m.s. error with iteration, (d) comparison of residuals in anomaly observed in the present method and Cordell's (1973) method.

magnitude of error (between the observed and model gravity anomalies) noticed in the present method with the one observed in Cordell's (1973) interpretation.

The inferred basement topography of the graben from the present inversion shows progressive deepening towards the west as observed in case of automatic modeling (Figure 5.7b). The maximum thickness of the graben estimated from the present inversion was 2.86 km, whereas Cordell reported a figure of 2.4 km (Figure 6.3b). The deciphered structure of the graben from automatic modeling is also shown in Figure 6.3b for comparison.

In short, the gravity anomalies of the San Jacinto graben analyzed by both automatic modeling and inversion yield more or less similar structural configuration with the estimated depths comparable to each other (Figure 6.3b).

6.5 Results and Discussion

Based on the principles of inversion, a space domain based algorithm and related software are developed to automatically interpret the gravity anomalies attributed to sedimentary basins among which the density contrast decreases with depth following a prescribed exponential form.

The noisy gravity anomalies of a synthetic model of a sedimentary basin were analyzed using the present software in the light of proposed inversion and found that the estimated structure almost coincides with the assumed structure. All the discontinuities present in the basement because of several synthetic and antithetic faults were also recovered successfully.

Further, the analysis of gravity anomalies of the San Jacinto graben, California using the proposed inversion has yielded a structural solution that is marginally deviated from the Cordell's (1973) interpreted model. The close fit between the observed and modeled gravity anomalies of the graben observed in the proposed method when compared to Cordell's (1973) interpretation has clearly demonstrates that the structural solution obtained from the present inversion is relatively more reliable than the one proposed by Cordell (1973).

It is of paramount interest to note that the estimated models from automatic modeling and inversion closely mimic with each other in both synthetic and real field examples. Further, from the analysis of gravity anomalies of synthetic and real field gravity anomalies it is found that the inversion algorithm performs lesser number of iterations in comparison to automatic modeling.

CHAPTER

SEVEN

Conclusions

The thesis consists of seven chapters. The novel features and important conclusions originate from the research work presented in Chapters II, III, IV, V, and VI are mainly on the following lines

1. Novelty in the interpretation methodologies including those of new forward modeling schemes,
2. Efficiency of an automatic inversion algorithm and related GUI software to analyze the gravity anomalies produced by 2D listric fault morphologies using a predefined exponential density contrast model,
3. Efficacy of an interactive modeling scheme to interpret the gravity anomalies of 2.5D listric fault morphologies using arbitrarily varying density contrast-depth models and related software,
4. Efficiency of an inversion algorithm to analyze the gravity anomalies for simultaneously estimating the density/depth parameters and fault plane geometries of 2.5D listric fault morphologies and related software,

5. Efficacies of automatic modeling and inversion strategies to analyze the gravity anomalies of 2D density interfaces using prescribed exponential density contrast models and related softwares'.

New forward modeling strategies combining both analytical and numerical approaches have been formulated in the spatial domain to compute the gravity anomalies of geologic structures among which the density contrast obeys exponential decrease with depth. The reliability of the forward modeling schemes are demonstrated on a few selected geophysical geometries by comparing the anomalies realized from the proposed methods with those obtained from the analytical methods.

Based on the interpretation methodologies presented in the thesis, a total of five new softwares' have been developed using the JAVA programming language to analyze the gravity anomalies for respective subsurface geologic structures. The inclusion of GUI makes the softwares more elegant and easy to operate. The novelty of the softwares is that besides generating the output in ASCII and graphical formats, each one displays the animated versions of the subsurface model growth and corresponding improvement in gravity response of the structure with iteration. The variations in the data misfit and density contrast shall also be displayed in animated forms in respective panels. Furthermore, the 2.5D algorithms presented in Chapters III and IV are fairly applicable to analyze the gravity anomalies even when the profiles run at an offset across the source(s).

The effects of strike length and offset of the profile on the magnitude of gravity anomaly is discussed in length in Chapter-III. It is concluded that these parameters should be considered invariably in the analysis of gravity anomalies to achieve reliable interpretations.

The applicability of each technique is exemplified with synthetic as well as real field gravity anomalies. In case of synthetic examples, pseudorandom noise was added to the gravity response of respective structures before being subjected to analysis. In all cases, the interpreted results are compared excellently well with the assumed parameters in case of synthetic examples and with the drilling/available information in case of field examples.

Scope for future research

An important problem to be addressed in gravity modeling studies is to develop new interpretation strategies, which shall take into account the presence of interfering sources within the sedimentary pack. The methodology elucidated in Chapter-VI can be extended to design appropriate inversion strategies to address the said problem.

An equally unsolved problem in gravity interpretation is to estimate the source parameters and regional gravity background simultaneously from a set of observed Bouguer gravity anomalies. Although, a few techniques are in vogue in this direction, each one has its own merits and demerits in its application. Therefore, it is expected that future research will contemplate more sophisticated regional-residual separation techniques, possibly integrated with the inversion processes described in the thesis, to cater more lucid interpretations of gravity anomalies.

In case of strike limited listric fault morphologies, the techniques presented in Chapters-III and IV presume that the detached hanging wall systems consist of several geologic formations; with each one bounded on the top and bottom by flat surfaces, which in reality may or not be so. The proposed techniques are more effective when the assumptions are relatively valid. It is anticipated that future research shall consider imperfect polynomial limiting surface for the fault plane and uneven bounding surfaces for the formations within the hanging wall systems to develop appropriate inversion schemes.

*****

```java
package com.ingrexp.view;

import java.awt.Frame;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

import com.ingrexp.control.INGREXP_Controller;
import com.ingrexp.model.INGREXP_CalculateValues;


public class INGREXP_MainView extends Frame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static void main(String s[])
    {
        INGREXP_MainView cm = new INGREXP_MainView();
        cm.setSize(1280, 768);

        cm.setTitle("INGREXP");
        cm.setResizable(false);
        cm.add(new INGREXP_MainPanel());
        cm.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit INGREXP ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(INGREXP_Controller.success==false){
                        String fileName = INGREXP_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        });
        INGREXP_MainPanel.img.addMouseListener(new MouseAction());
        cm.setVisible(true);
    }
}

class MouseAction extends MouseAdapter{
    public void mousePressed(MouseEvent e) {

        INGREXP_CalculateValues.drawGraph();
    }

}
-------------------------------------------------------------------------------------------
```

```java
package com.ingrexp.view;

import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextArea;
```

```java
import java.awt.TextField;

import java.io.File;
import java.io.IOException;
import java.util.HashMap;

import javax.swing.JFileChooser;

import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

import com.ingrexp.view.INGREXP_MainPanel;

public class INGREXP_MainPanel extends Panel  {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static TextArea img = new TextArea(36,140);
    Panel p_North, p_West;
    public static Panel p_East;

    static Panel p_South;

    public static Panel p_Center;
    static TextField inputValues [] = new TextField[12];

    Button actionButton[] = new Button[6];
    String rowdata[][]={};

    /**Field Area Name*/
    final static int AREA_FE = 0;
    /**Number of the Profile*/
    final static int NUM_PROFILE =  1;
    /**Number of observations*/
    final static int N_OBS = 2 ;
    /**Distance(km)*/
    final static int X_KM = 3;
    /**Elevation(km)*/
    final static int ELE_KM = 4;
    /**observed anomalies*/
    final static int NOB_GOB = 5;
    /** Surface density contrast (gm/cc) */
    final static int SD_POLY = 6;
    /**Lambda*/
    final static int LAMBDA_ST = 7 ;
    /**Number of iteration values*/
    final static int NOB_ITER = 8;

    public INGREXP_MainPanel(){

        this.setLayout(new BorderLayout());
        p_North = new Panel();
        p_West = new Panel();
        p_East = new Panel ();
        p_South = new Panel();
        p_Center = new Panel();

        Label graphLabel = new Label("INVERSION OF GRAVITY ANOMALIES OF 2D LISTRIC FAULT 
STRUCTURES USING EXPONENTIAL DENSITY MODEL", Label.CENTER);
        graphLabel.setFont(new Font("Bold", 1, 15));
        p_Center.add(graphLabel);

        for(int i = 0; i < 9; i++){
            inputValues[i] = new TextField();
        }
        p_North.setFont(new Font("Bold",1,12));
        actionButton[0] = new Button("Load data");
        actionButton[1] = new Button("Interpretation");
        actionButton[2] = new Button("Save & Print");
        actionButton[3] = new Button("Clear");
```

```
            actionButton[4] = new Button("Exit");

            this.populateNorthPanel();
            INGREXP_TableView.populateEastPanel(rowdata);
            this.add(p_North, BorderLayout.NORTH);
            p_Center.setSize(1000, 760);
            this.add(p_Center, BorderLayout.CENTER);
            img.setEditable(false);
            p_Center.add(img);
            this.add(p_East, BorderLayout.EAST);
            this.setVisible(true);
        }

        public void populateNorthPanel(){
            p_North.setLayout(new GridLayout(4,6));
            p_North.add(new Label("Area Name"));
            p_North.add(inputValues[0]);
            p_North.add(new Label("Profile Name"));
            p_North.add(inputValues[1]);
            p_North.add(new Label("Number of observations"));
            p_North.add(inputValues[2]);
            p_North.add(new Label("Distance (km)"));
            p_North.add(inputValues[3]);
            p_North.add(new Label("Elevation of each station (km)"));
            p_North.add(inputValues[4]);
            p_North.add(new Label("Observed anomalies (mGal)"));
            p_North.add(inputValues[5]);
            p_North.add(new Label("Surface density contrast (gm/cc)"));
            p_North.add(inputValues[6]);
            p_North.add(new Label("Lambda (1/km)"));
            p_North.add(inputValues[7]);
            p_North.add(new Label("Iterations"));
            p_North.add(inputValues[8]);

            p_North.add(new Label(""));
            p_North.add(actionButton[0]);
            p_North.add(actionButton[1]);
            p_North.add(actionButton[2]);
            p_North.add(actionButton[3]);
            p_North.add(actionButton[4]);

            actionButton[0].addActionListener(new com.ingrexp.control.INGREXP_Controller());
            actionButton[1].addActionListener(new com.ingrexp.control.INGREXP_Controller());
            actionButton[2].addActionListener(new com.ingrexp.control.INGREXP_Controller());
            actionButton[3].addActionListener(new com.ingrexp.control.INGREXP_Controller());
            actionButton[4].addActionListener(new com.ingrexp.control.INGREXP_Controller());

        }

        public static HashMap captureValues(){
            HashMap h_Map = new HashMap();
            try {
                h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
                h_Map.put("NUM_PROFILE", inputValues[NUM_PROFILE].getText());
                h_Map.put("N_OBS", inputValues[N_OBS].getText());
                h_Map.put("X_KM", inputValues[X_KM].getText());
                h_Map.put("ELE_KM", inputValues[ELE_KM].getText());
                h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
                h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
                h_Map.put("ALPHA_ST",inputValues[LAMBDA_ST].getText());
                h_Map.put("NOB_ITER", inputValues[NOB_ITER].getText());

            }
            catch (Exception e) {
                e.printStackTrace();
            }
            return h_Map;

        }

        public static void clearPanel(Panel p) {
            Graphics g = p.getGraphics();
            g.setColor(Color.WHITE);
            g.fillRect(0, 30, 1280, 650);
        }
```

```
    public static void loadData1()throws IOException  {
        try{

            String current = System.getProperty("user.dir");
            JFileChooser chooser=new  JFileChooser(current);
            int returnVal = chooser.showOpenDialog(null);
            String dis[], ele[],gobs[];
            String disval = "" ,eleval="",gobsval="";
            Workbook w;

            if(returnVal == JFileChooser.APPROVE_OPTION) {
                File f = chooser.getSelectedFile();
                w = Workbook.getWorkbook(f);
                Sheet sheet = w.getSheet(0);
                dis = new String[sheet.getRows()+1];
                ele = new String[sheet.getRows()+1];
                gobs = new String[sheet.getRows()+1];
                for (int j = 0; j < sheet.getColumns(); j++) {
                    for (int i = 1; i < sheet.getRows(); i++) {
                        Cell cell = sheet.getCell(j, i);
                        CellType type = cell.getType();
                        if (type == CellType.LABEL) {
                            INGREXP_MainPanel.inputValues[INGREXP_MainPanel.AREA_FE].setTex
(cell.getContents());
                        }
                        if (type == CellType.NUMBER) {
                            if (j == 1){
                                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.NUM_PROFILE
.setText(cell.getContents());
                            }
                            if (j == 2){
                                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.N_OBS].setT
xt(cell.getContents());
                            }
                            if (j == 3){
                                dis[i] = cell.getContents()+",";
                                disval = disval + dis[i];
                            }
                            if (j == 4){
                                ele[i] = cell.getContents()+",";
                                eleval = eleval + ele[i];
                            }
                            if (j == 5){
                                gobs[i] = cell.getContents()+",";
                                gobsval = gobsval + gobs[i];
                            }
                            if (j == 6){
                                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.SD_POLY].se
Text(cell.getContents());
                            }
                            if (j == 7){
                                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.LAMBDA_ST].
etText(cell.getContents());
                            }
                            if (j == 8){
                                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.NOB_ITER].s
tText(cell.getContents());
                            }
                        }
                    }
                }
                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.X_KM].setText(""+disval);

                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.ELE_KM].setText(""+eleval);
                INGREXP_MainPanel.inputValues[INGREXP_MainPanel.NOB_GOB].setText(""+gobsva
;
            }
        }
        catch (BiffException e) {
            e.printStackTrace();
        }
    }
```

```java
    public static void clearDefaultValues(){
        inputValues[AREA_FE].setText("");
        inputValues[NUM_PROFILE].setText("");
        inputValues[N_OBS].setText("");
        inputValues[X_KM].setText("");
        inputValues[ELE_KM].setText("");
        inputValues[NOB_GOB].setText("");
        inputValues[SD_POLY].setText("");
        inputValues[LAMBDA_ST].setText("");
        inputValues[NOB_ITER].setText("");
    }
}
------------------------------------------------------------------------------------------

package com.ingrexp.view;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Panel;
import java.awt.TextArea;

import javax.swing.JScrollPane;
import javax.swing.JTable;

import com.ingrexp.view.INGREXP_MainPanel;

public class INGREXP_TableView extends Panel{


    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static TextArea val = new TextArea(5,5);
    public static  void populateEastPanel(Object rowData[][]) {
        com.ingrexp.view.INGREXP_MainPanel.p_East.removeAll();

        com.ingrexp.view.INGREXP_MainPanel.p_East.setLayout(new GridLayout(2,1));


        Object columnNames[] = {"Distance(km)", "Observed anomalies (mGal)", "Calculated
anomalies (mGal)", "Error (mGal)"};
        JTable table = new JTable(rowData, columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(300,500));
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setAutoscrolls(true);

        com.ingrexp.view.INGREXP_MainPanel.p_East.add(scrollPane);
        val.setEditable(false);
        com.ingrexp.view.INGREXP_MainPanel.p_East.add(val);
        try{
            com.ingrexp.view.INGREXP_MainPanel.p_East.validate();
        }
        catch(Exception e){

            Graphics g =INGREXP_MainPanel.img.getGraphics();
            g.setColor(Color.white);
            g.fillRect(0, 0, 1000, 600);
            g.setColor(Color.black);
            g.setFont(new Font("Arial", 20, 40));
            g.drawString("ERROR...", 300, 400);
        }
        com.ingrexp.view.INGREXP_MainPanel.p_East.setVisible(true);
    }
}
------------------------------------------------------------------------------------------

package com.ingrexp.view;

import java.applet.Applet;
```

```java
import java.awt.Color;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics2D;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.text.DecimalFormat;



import com.ingrexp.model.INGREXP_CalculateValues;
import com.ingrexp.util.INGREXP_Utility;

public class INGREXP_DrawGraph extends Applet{
    private static final long serialVersionUID = 1L;
    public static int i_no_obs;
    float maxY,maxZ,maxX ;
    double obs[];
    double inidep;
    public void drawGraph(Graphics2D g2) {

        g2.setFont(new Font("Arial", 20, 12));
        g2.setColor(Color.BLACK);
        g2.draw(new Line2D.Float(150, 50, 150, 300));
        g2.drawLine(90,35 ,1040,35);
        g2.drawLine(780, 35, 780, 560);
        g2.drawLine(90, 560, 1040, 560);
        g2.drawLine(1040, 35, 1040, 560);
        g2.drawLine(90, 35, 90, 560);

        String []a = {"A", "N", "O", "M", "A", "L", "Y", "(m", "G", "a", "l", "s)"};
        String []b = {"D", "E", "P", "T", "H", "(k", "m)"};
        for (int i = 0; i < a.length; i++) {
            g2.drawString(""+a[i], 100, 20 + 60 + ( i * 20 ) );
        }
        for (int i = 0; i < b.length; i++) {
            g2.drawString(""+b[i], 100, 20 + 350 + ( i * 20 ) );
        }

    }

    public void plot(Graphics2D g2) {

        g2.setFont( new Font("Arial", 12, 12) );
        DecimalFormat f = new DecimalFormat("0.#");
        g2.setColor(Color.BLACK);

        i_no_obs = INGREXP_CalculateValues.input_n_obs;
        inidep  = INGREXP_CalculateValues.o_par[1];
        obs = new double[i_no_obs+1];
        for (int i = 1; i <= i_no_obs; i++) {
            obs[i] = INGREXP_CalculateValues.input_x_km[i];
        }

        maxX = (float)obs[i_no_obs];
        float maxy = (float)
INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.input_nob_gob);
        float maxy1 = (float)
INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.o_GC);
        if(maxy > maxy1)
            maxY = maxy;
        else
            maxY = maxy1;
        maxZ = (float)INGREXP_CalculateValues.o_par[1];

        g2.drawString("|",(float) 600  , 308);
        g2.drawString(""+f.format(INGREXP_CalculateValues.input_x_km[i_no_obs]),(float) 600
, 323);
        g2.drawString("0", 125, 310);
        g2.drawString("DISTANCE(km)", 315, 295);
        float xplot = 0;
        float xInterval = (float) (INGREXP_CalculateValues.input_x_km[i_no_obs] / 5);

        int zInterval = 50;
        for (float x = xInterval, j = 1; x < 600; x += xInterval){
```

```
                xplot = xplot + xInterval;
                if(j > 4)
                    break;
                g2.drawString("|",(float) (150 + (450 * x / maxX) ), 308);
                g2.drawString("" + f.format(xplot), (float) (150 + (450 * x / maxX) ) - 3, 323)
                j++;
            }

            DecimalFormat d =  new DecimalFormat("0.#");
            float points1 = maxZ / 5 ;
            for (int x = zInterval + 250,j = 1; x < 550; x += zInterval){
                g2.drawString("-", 148, 52 + x);
                g2.drawString("" +d.format(points1 * j), 125, 50 + x);
                j++;
            }
        }


        public void plotXYCoordinates(Graphics2D g2){

            double minAno =
    INGREXP_Utility.findMinimumNumber1(INGREXP_CalculateValues.input_nob_gob);
            double maxAno =
    INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.input_nob_gob, minAno);
            double minObAno = INGREXP_Utility.findMinimumNumber1(INGREXP_CalculateValues.o_GC)
            double maxObAno = INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.o_GC,
    minObAno);

            if (minAno < 0 && maxObAno < 0 && maxAno < 0 && minObAno <0 ){
                plotXYCoordinates1(g2);
            }
            if (minAno >= 0 && maxObAno > 0){
                plotXYCoordinates1(g2);
            }
            if (minAno < 0 && maxObAno > 0 || maxAno>0 && minObAno<0){
                plotXYCoordinates2(g2);
            }
        }

        public void plotXYCoordinates1 (Graphics2D g2) {

            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.black);
            float maxval = (float)
    INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.o_GC);
            float maxval1 = (float)
    INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.input_nob_gob);
            if(Math.abs(maxval)>Math.abs(maxval1))
                maxY = maxval;
            else
                maxY = maxval1;
            int points = (int)maxY / 5;
            int yInterval = 50;
            g2.drawString("0", 125, 50);
            for (int x = yInterval, j = 1; x < 250; x += yInterval){

                g2.drawString("-", 148, 50 + x);
                g2.drawString("" + (points*j), 125, 50 + x);
                j++;
            }
            float prevx = (float) (150 +( 450 * obs[1] / maxX));
            float prevy = (float)( 50 + ( 250 * INGREXP_CalculateValues.o_GC[1] / maxY ) );
            float xpoint = 0;
            float ypoint = 0;
            float gypoint = 0;

            for (int k = 1; k <= i_no_obs; k++) {
                xpoint = (float)( 450 * obs[k] / maxX);
                ypoint = (float)( ( 250 * INGREXP_CalculateValues.o_GC[k] / maxY ) );
                gypoint = (float)( ( 250 * INGREXP_CalculateValues.input_nob_gob[k] / maxY ) )
                g2.setColor(Color.BLACK);
                g2.draw(new Line2D.Float(prevx, prevy, 150+ xpoint, 50 + ypoint ));
                g2.setColor(Color.BLUE);
                g2.setFont(new Font("Arial", 20, 40));
                g2.drawString(".",  150+xpoint - 6 , 50 + gypoint + 3);
```

```
                g2.setFont(new Font("Arial", 20, 12));
                g2.setColor(Color.black);
                prevx = 150 + xpoint;
                prevy = 50 + ypoint ;
            }
        }


        public void plotXYCoordinates2 (Graphics2D g2) {
            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.black);
            int countPosObs=0,countNegObs=0,countPosCal=0,countNegCal=0;
            double store[] = new double[INGREXP_CalculateValues.input_n_obs + 1];
            double store1[] = new double[INGREXP_CalculateValues.input_n_obs + 1];
            double negStore[] = new double[INGREXP_CalculateValues.input_n_obs + 1];
            double negStore1[] = new double[INGREXP_CalculateValues.input_n_obs + 1];
            for(int i = 1; i <= INGREXP_CalculateValues.input_n_obs; i++){
                if(INGREXP_CalculateValues.o_GC[i] > 0){
                    store[i] = INGREXP_CalculateValues.o_GC[i];
                    countPosCal = countPosCal + 1;
                }
                else{
                    negStore[i] = INGREXP_CalculateValues.o_GC[i];
                    countNegCal = countNegCal + 1;
                }
                if(INGREXP_CalculateValues.input_nob_gob[i]>0){
                    store1[i] = INGREXP_CalculateValues.input_nob_gob[i];
                    countPosObs = countPosObs + 1;
                }
                else{
                    negStore1[i] = INGREXP_CalculateValues.input_nob_gob[i];
                    countNegObs = countNegObs + 1;
                }
            }

            float maxPos = (float) INGREXP_Utility.findMaximumNumber1(store);
            float maxPos1 = (float) INGREXP_Utility.findMaximumNumber1(store1);
            float maxNeg = (float) INGREXP_Utility.findMaximumNumber(negStore);
            float maxNeg1 = (float) INGREXP_Utility.findMaximumNumber(negStore1);
            float posNum =0;

            if(maxPos > maxPos1)
                posNum = maxPos;
            else
                posNum = maxPos1;
            if(Math.abs(maxNeg) > Math.abs(maxNeg1))
                maxY = maxNeg;
            else
                maxY = maxNeg1;

            float prevx = (float) (150 +( 450 * obs[1] / maxX));;
            float prevy = 0;
            if(INGREXP_CalculateValues.o_GC[1]>0){
                if(countNegCal > countPosObs)
                    prevy = 100 - (float)( ( 50 * INGREXP_CalculateValues.o_GC[1] / posNum ) );
                else
                    prevy = 200 - (float)( ( 150 * INGREXP_CalculateValues.o_GC[1] / posNum ) )
            }
            else{
                if(countNegCal > countPosObs)
                    prevy = 100 + (float)( ( 200 * INGREXP_CalculateValues.o_GC[1] / maxY ) );
                else{
                    prevy = 200+(float)( ( 100 * INGREXP_CalculateValues.o_GC[1] / posNum ) );
                }
            }
            float xpoint = 0;
            float ypoint = 0;
            float gypoint = 0;
            float points = 0;

            DecimalFormat f = new DecimalFormat("0.#");
            if(countNegCal > countPosObs){
                g2.drawString("-", 148, 100);
                g2.drawString("0",125,100);
            }
```

```java
            else{
                g2.drawString("-", 148, 200);
                g2.drawString("0",125 ,200);
            }
            g2.drawString("-", 148, 55);
            g2.drawString(""+f.format(posNum), 125, 55);

            if(countNegCal > countPosObs){
                points = maxY / 4;
                int yInterval=50;
                for (int x = yInterval, j = 1; x < 250; x+=yInterval){

                    g2.drawString("-", 148, 100 + x );
                    g2.drawString("" + f.format(points * j), 125, 100 + x );
                    j++;
                }
            }
            else{
                points = posNum / 3;
                int yInterval=50;
                for (int x = yInterval, j = 1; x < 200; x+=yInterval){

                    g2.drawString("-", 148, 205 - x );
                    g2.drawString("" + f.format(points * j), 125, 205 - x );
                    j++;
                }

                g2.drawString("-", 148, 250 );
                g2.drawString("-" + f.format(posNum/2), 125 , 250 );
                g2.drawString("-", 148, 300 );
                g2.drawString("-" + f.format(posNum), 125, 300 );

            }
            for (int k = 1; k <= i_no_obs; k++) {

                xpoint = (float)( 450 * obs[k] / maxX);
                if(INGREXP_CalculateValues.o_GC[k]>0){
                    if(countNegCal > countPosObs)
                        ypoint = 100-(float)( ( 50 * INGREXP_CalculateValues.o_GC[k] / posNum )
    );
                    else
                        ypoint = 200-(float)( ( 150 * INGREXP_CalculateValues.o_GC[k] / posNum
    );
                }
                else{
                    if(countNegCal > countPosObs)
                        ypoint = 100+(float)( ( 200 * INGREXP_CalculateValues.o_GC[k] / maxY )
    );
                    else
                        ypoint = 200+(float)( ( 100 * INGREXP_CalculateValues.o_GC[k] / posNum
    );
                }
                if(INGREXP_CalculateValues.input_nob_gob[k]>0){
                    if(countNegCal > countPosObs)
                        gypoint = 100-(float)( ( 50 * INGREXP_CalculateValues.input_nob_gob[k]
    posNum ) );
                    else
                        gypoint = 200-(float)( ( 150 * INGREXP_CalculateValues.input_nob_gob[k]
    / posNum ) );
                }
                else{
                    if(countNegCal > countPosObs)
                        gypoint = 100+(float)( ( 200 * INGREXP_CalculateValues.input_nob_gob[k]
    / maxY ) );
                    else
                        gypoint = 200+(float)( ( 100 * INGREXP_CalculateValues.input_nob_gob[k]
    / posNum ) );
                }
                g2.setColor(Color.BLACK);
                g2.draw(new Line2D.Float(prevx, prevy, 150 + xpoint,  ypoint  ));

                g2.setColor(Color.BLUE);
                g2.setFont(new Font("Arial", 20, 40));
                g2.drawString(".",  150+xpoint - 6 ,  gypoint+3 );

```

```java
                g2.setFont(new Font("Arial", 20, 12));
                g2.setColor(Color.black);
                prevx = 150 + xpoint;
                prevy = ypoint ;

            }
    }
    public void drawOBJ(Graphics2D g2) {

        g2.setColor(Color.BLACK);
        g2.drawLine(820, 70, 820, 160);
        g2.drawLine(820, 160, 910, 160);
        g2.drawString("J", 800, 90);

        double maxOb = INGREXP_Utility.findMaximumNumber1(INGREXP_CalculateValues.o_funct);
        int ini = INGREXP_Utility.findMaximumNumber(INGREXP_CalculateValues.o_iter);
        if(ini == 5)
            ini= ini + 1;
        int maxiter = ( ini / 3 * 5 ) * 2;
        int point;
        int xInterval = 22;
        point = ( ( ini ) / 3 * 5 ) / 5;

        for (int x = xInterval, j = 1; x < 90 ; x += xInterval) {

            g2.drawString("'", 821 + x, 170);
            g2.drawString("" + (point*j), 820 + x-3, 175);
            j++;
        }

        float prevx = 820;
        float prevy = 70;
        float xpoint = 0;
        float ypoint = 0;

        for (int i = 1; i <= INGREXP_CalculateValues.o_iter; i++) {

            xpoint = (float)( 250 * i /maxiter );
            ypoint = 70 - (float)  ( ( 90 * (INGREXP_CalculateValues.o_funct[i]) / maxOb ) );

            if(i==INGREXP_CalculateValues.o_iter){
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint-4, 90 + ypoint));
            }
            else {
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 90 + ypoint));
            }
            prevx = 820 + xpoint;
            prevy = 90 + ypoint;

        }
        DecimalFormat d1= new DecimalFormat("0.###");
        DecimalFormat d= new DecimalFormat("0.#");
        g2.drawString(" "+d.format(INGREXP_CalculateValues.o_funct[1]), 780, 70);
        g2.drawString(" "+d1.format(INGREXP_CalculateValues.o_funct[INGREXP_CalculateValues.o_iter]), 820 + xpoint, 90 + ypoint);
        g2.setFont(new Font("Arial", 40,11));
        g2.drawString ("Iterations",850,186);
    }

    public void drawSd(Graphics2D g2) {
        g2.setColor(Color.black);
        g2.drawLine(780, 200, 1040, 200);
        g2.setColor(Color.red);
        g2.setFont(new Font("Arial", 20, 12));
        DecimalFormat d= new DecimalFormat("0.##");
        DecimalFormat d2= new DecimalFormat("0.#");
        DecimalFormat d1= new DecimalFormat("0.###");

        g2.drawString(""+d.format(inidep), 790, 550);
        g2.drawString("-", 820, 552);
        g2.drawString("0", 807 , 300);
        g2.drawLine(820, 300, 910, 300);
        g2.draw(new Line2D.Float(820, 300, 820,  550));
```

```java

            double maxOb1 = INGREXP_Utility.findMaximumNumber1(INGREXP_CalculateValues.vsd);
            float points = maxZ / 5 ;
            int zInterval = 50;
            for(int x = zInterval+250,j = 1; x < 550; x+=zInterval){
                if(j>4)
                    break;
                g2.drawString("-", 820, 50 + x + 2);
                g2.drawString("" +d2.format(points * j), 790, 50 + x);
                j++;
            }
            float prevx = 820+ (float) ( ( 90 * ( Math.abs(INGREXP_CalculateValues.vsd[1] )) /
    maxOb1 ) );
            float prevy = 300;
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= INGREXP_CalculateValues.count; i++) {
                xpoint = (float)( 90 * Math.abs(INGREXP_CalculateValues.vsd[i]) / maxOb1 );
                ypoint = (float)( 250 * INGREXP_CalculateValues.dep[i] / maxZ );
                g2.setColor(Color.blue);
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 300 + ypoint));

                prevx = 820 + xpoint;
                prevy = 300 + ypoint;
            }

            g2.drawString(""+d.format(INGREXP_CalculateValues.vsd[1] ),805+ (float) ( ( 90 * (
    Math.abs(INGREXP_CalculateValues.vsd[1] )) / maxOb1 ) ) ,300 );
            g2.drawString(""+d1.format(INGREXP_CalculateValues.vsd[INGREXP_CalculateValues.cour
    ] ),820+ (float) ( ( 90 * (
    Math.abs(INGREXP_CalculateValues.vsd[INGREXP_CalculateValues.count] )) / maxOb1 ) ),
    300+(float)( 250 * inidep / maxZ ) );
            g2.setColor(Color.BLACK);
            g2.drawString("Variation of density contrast " , 800,220);
            g2.drawString("with depth" , 850,240);
            g2.setFont(new Font("Arial", 40,11));
            g2.drawString ("Density contrast",830,285);
            g2.drawString ("(gm/cc)",843,295);
            g2.drawString("Z(km)", 790,(float)( 300+((250*inidep/maxZ))/2));

        }
        public void plotZCoordinates (Graphics2D g2) {

            GradientPaint gradient = new GradientPaint(10, 10, Color.yellow, 30, 200,
    Color.MAGENTA, true);
            g2.setPaint(gradient);
            g2.fill(new Rectangle2D.Float(150, 300, 450 , 250));
            g2.setColor(Color.BLACK);
            i_no_obs = INGREXP_CalculateValues.input_n_obs;
            obs = new double[i_no_obs+1];
            for (int i = 1; i <= i_no_obs; i++) {
                obs[i] = INGREXP_CalculateValues.input_x_km[i];
            }
            maxX = (float) obs[i_no_obs];
            maxZ = (float)INGREXP_CalculateValues.o_par[1];
            float s = 0;
            float fc = 0;
            float spoint = 300;
            float fcpoint = (float)( 150 + ( 450 * INGREXP_CalculateValues.o_par[2] / maxX ) );
            float xpoint = 0;
            float zpoint = 0;
            while (s <= INGREXP_CalculateValues.o_par[1]) {
                float z1 = (float) 0.001;
                s = s + z1;
                fc = 0;
                for (int i = 1; i<4; i++){
                    fc = (float) (fc + INGREXP_CalculateValues.o_par[i + 1] * Math.pow(s, i -
    1));
                }
                xpoint = (float)( 450 * fc / maxX);
                zpoint = (float)( 250 * s / maxZ);
                g2.setColor(Color.BLACK);
                g2.draw(new Line2D.Float(fcpoint, spoint,(float)150 + xpoint, 300 + zpoint));
                g2.setColor(Color.RED);
```

```
                g2.draw(new Line2D.Float(150, 300 + zpoint, (float) 150 + xpoint, 300 + zpoint));
                fcpoint = (float) 150 + xpoint;
                spoint = 300 + zpoint;
            }
            g2.setColor(Color.BLACK);
            g2.drawLine(150, 300, 600, 300);
            g2.drawLine(150, 300, 550, 300);
        }
        public void idex(Graphics2D g){
            g.setColor(Color.BLUE);
            g.setFont(new Font("Arial", 20, 50));
            g.drawString(" ...     ",595,70);
            g.setFont(new Font("Arial", 20, 12));
            g.drawString("Observed anomalies",650,70);
            g.setColor(Color.BLACK);
            g.drawString("____:",615,87);
            g.drawString("Calculated anomalies",650,90);
            GradientPaint gradient = new GradientPaint(10, 10, Color.yellow, 30, 50, Color.MAGENTA, true);
            g.setPaint(gradient);
            g.fillRect(615, 100, 35, 10);
            g.setColor(Color.BLACK);
            g.drawString(": Estimated Depth", 650, 110);
            g.drawString("Structure", 660, 130);
        }
    }
    ------------------------------------------------------------------------------------------

    package com.ingrexp.model;

    import java.awt.Color;
    import java.awt.Graphics;
    import java.awt.Graphics2D;
    import java.awt.event.MouseAdapter;
    import java.awt.event.MouseEvent;
    import java.awt.event.MouseListener;
    import java.awt.image.BufferedImage;
    import java.io.File;
    import java.io.FileOutputStream;
    import java.text.DecimalFormat;
    import java.util.HashMap;

    import javax.imageio.ImageIO;
    import com.ingrexp.model.INGREXP_NOREQ;

    import com.ingrexp.util.INGREXP_Utility;
    import com.ingrexp.view.INGREXP_MainPanel;
    import com.ingrexp.view.INGREXP_TableView;

    public class INGREXP_CalculateValues {
        public static int input_n_obs = 0;
        public static double input_x_km[];
        public static double input_nob_gob[];
        public static Object obj[][] = null;
        public static double []o_GC ;
        public static double []o_err ;
        public static double []o_par;
        public static double []o_funct ;
        public static double []vsd = null;
        public static double []dep = null;
        public static double o_func;
        public static int o_iter,count;
        public static String input_area_name,input_profile_num="";
        public static int np;
        public static BufferedImage image;

        public void getAnamolyValues(HashMap h_Map)   {
            double ALER = 0.0000001;
            double input_sd_poly = 0;
            double input_lambda_val = 0;
            int input_ndeg = 2;
            double []input_ele_km = null;
            int input_nob_iter = 0;

```

```java


        try {

            input_n_obs = INGREXP_Utility.convertInteger((String)h_Map.get("N_OBS"));
            input_x_km = INGREXP_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
            input_ele_km = INGREXP_Utility.convertDoubleArray((String)h_Map.get("ELE_KM"));
            input_nob_gob = INGREXP_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
            input_sd_poly = INGREXP_Utility.convertDouble((String)h_Map.get("SD_POLY"));
            input_lambda_val = INGREXP_Utility.convertDouble((String)h_Map.get("ALPHA_ST"));;
            input_nob_iter = INGREXP_Utility.convertInteger((String)h_Map.get("NOB_ITER"));
            input_area_name = INGREXP_Utility.convertString((String)h_Map.get("AREA_FE"));
            input_profile_num = INGREXP_Utility.convertString((String)h_Map.get("NUM_PROFILE"));
        }
        catch(Exception e) {
            e.printStackTrace();
        }

        o_GC  = new double[input_n_obs + 1];
        o_err = new double[input_n_obs + 1];
        o_par = new double[input_n_obs + 1];
        o_funct  = new double[input_nob_iter + 1];

        double []g1 = new double[input_n_obs + 1];
        double []g2 = new double[input_n_obs + 1];
        double gc[] = new double[input_n_obs + 1];
        double err[]= new double[input_n_obs + 1];

        double [][]p = new double[8][9];
        double [][]s = new double[8][input_n_obs + 1];
        double []b = new double[8];

        double []par = new double[8];
        double []par1 = new double[8];
        double []par2 = new double[8];
        double []dupar = new double[8];

        double []KS = new double[2];
        double funct2 = 0;
        double lambda = 0.5;
        np = input_ndeg + 2;
        double gmax = Math.abs(input_nob_gob[1]);

        for (int k = 1; k <= input_n_obs; k++) {
            if (Math.abs(input_nob_gob[k]) - gmax > 0)
                gmax = Math.abs(input_nob_gob[k]);
        }

        double datum = input_nob_gob[1];
        double r = input_nob_gob[input_n_obs] - input_nob_gob[1];
        int kk = 1;
        double gh = 0.5 * r;
        kk = kk + 1;
        double XH = 0;

        while ((( input_nob_gob[kk] - datum) / gh ) - 1.0 < 0) {
            kk = kk + 1;
        }
        if ((( input_nob_gob[kk] - datum) / gh) - 1.0 > 0) {
            XH = input_x_km[kk-1] + ( ( gh + datum - input_nob_gob[kk-1]) * ( input_x_km[kk] - input_x_km[kk-1] ) ) / ( input_nob_gob[kk] - input_nob_gob[kk-1] );
        }
        if ((( input_nob_gob[kk] - datum ) / gh) - 1.0 == 0) {
            XH = input_x_km[kk];
        }

        double amax = - gmax;
        if(input_lambda_val == 0)
            par[1] = (amax) /  ( 13.3333 * 3.14159265 * input_sd_poly);
        else
            par[1] = (1 / input_lambda_val) * Math.log(1 + ((input_lambda_val * amax) / (13.3333 * 3.14159265 * input_sd_poly)));
```

```
            par[2] = XH;
            double dpar = 0.01;

            INGREXP_NOREQ.getGF (input_n_obs, np, input_ndeg, input_ele_km, input_sd_poly, input_lambda_val, input_x_km, par, gc);
            double funct1 = 0;
            for (int k = 1; k <= input_n_obs; k++) {
                err[k] = input_nob_gob[k] - gc[k];
                funct1 = funct1 + Math.pow( err[k] , 2 );
            }
            funct1 = Math.sqrt(funct1 / input_n_obs);
            lambda = 0.5;
            int NP1 = np + 1;
            int IER = 1;

            while (IER <= input_nob_iter) {
                int ITER1 = IER ;
                o_funct[ITER1] = funct1;
                for (int K = 1; K <= np; K++) {
                    par1[K] = par[K];
                }

                for (int I = 1; I <= np; I++) {
                    par1[I] = par[I] + dpar / 2.0;
                    INGREXP_NOREQ.getGF (input_n_obs, np, input_ndeg, input_ele_km, input_sd_poly, input_lambda_val, input_x_km, par1, g1);
                    par1[I] = par[I] - dpar / 2.0;
                    INGREXP_NOREQ.getGF (input_n_obs, np, input_ndeg, input_ele_km, input_sd_poly, input_lambda_val, input_x_km, par1, g2);
                    for (int K = 1; K <= input_n_obs; K++) {
                        s[I][K] = ( g1[K] - g2[K] ) / dpar;
                    }
                }
                for (int J = 1; J <= NP1; J++) {
                    for(int I = 1; I <= np; I++) {
                        p[I][J] = 0.0;
                    }
                }
                for (int J = 1; J <= np; J++) {
                    for (int I = 1; I <= np; I++) {
                        for (int K = 1; K <= input_n_obs; K++){
                            p[I][J] = p[I][J] + s[I][K] * s[J][K];
                        }
                    }
                }

                for (int J = 1; J <= np; J++) {
                    for (int K = 1;K <= input_n_obs; K++) {
                        p[J][NP1] = p[J][NP1] + err[K] * s[J][K];
                    }
                }

                do {
                    double con = lambda + 1.0;
                    for (int I = 1; I <= np; I++) {
                        dupar[I] = par[I];
                    }
                    for (int L = 1; L <= np; L++) {
                        for (int J = 1;J <= np; J++) {
                            if (L - J == 0)
                                p[L][J] = p[L][J] * con;
                        }
                    }
                    INGREXP_NOREQ.getNOREQ(p, b, np, KS);
                    for (int I = 1; I <= np; I++) {
                        par2[I] = dupar[I] +  b[I];
                    }

                    INGREXP_NOREQ.getGF (input_n_obs, np, input_ndeg, input_ele_km, input_sd_poly, input_lambda_val, input_x_km, par2, gc);
                    funct2 = 0.0;
                    for (int K = 1; K <= input_n_obs; K++) {
                        err[K] = input_nob_gob[K] - gc[K];
                        funct2 = funct2 + Math.pow(err[K] , 2);
                    }
```

```
                    funct2 = Math.sqrt(funct2 / input_n_obs);

                    if (funct1 - funct2 < 0) {
                        lambda = lambda * 2.0;
                        for (int i = 1; i <= np; i++) {
                            for (int j = 1; j <= np; j++) {
                                if (i - j == 0) {
                                    p[i][j] = p[i][j] / con;
                                }
                            }
                        }
                    }

                } while (funct1 - funct2 < 0);

                funct1 = funct2;
                IER++;
                for (int I = 1; I <= np; I++) {
                    par[I] = par2[I];
                }
                o_iter = ITER1;
                o_func = funct2;
                for (int K = 1;K <= input_n_obs; K++) {
                    o_GC[K] = gc[K];
                    o_err[K] = err[K];
                }
                for (int l = 1;l <= np; l++) {
                    o_par[l] = par[l];
                }

                if ( funct2 < ALER || ITER1 == input_nob_iter||lambda -12 > 0 ) {
                    o_iter = ITER1;
                    o_func = funct2;
                    for (int K = 1;K <= input_n_obs; K++) {
                        o_GC[K] = gc[K];
                        o_err[K] = err[K];
                    }
                    for (int l = 1;l <= np; l++) {
                        o_par[l] = par[l];
                    }

                    denCal(input_sd_poly,input_lambda_val);
                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC,
        o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }
                else{
                    denCal(input_sd_poly,input_lambda_val);
                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC,
        o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }

                //Thread.sleep(10);

                if ( funct2 < ALER||lambda -12 > 0)
                    break;

                lambda = lambda / 2.0;

            }
        }
        public static void denCal(double sd,double la){

            int i = 1;
            double z1 = 0.0001;
            double z2 = INGREXP_CalculateValues.o_par[1];
            vsd = new double[(int) Math.pow(input_n_obs, 2)];
            dep = new double[(int) Math.pow(input_n_obs, 2)];
            while(z1 <= z2){
                double dc =  sd * Math.exp(-la * z1);
                vsd[i] = dc;
                dep[i] = z1;
                z1 = z1+0.1;
                i++;
```

```java
            }
            count = i;
            vsd[count] = sd * Math.exp(-la * z2);
            dep[count] = z2;

        }

        public static void  setGraphValues(int i_no_obs, int np, int ite, double []dis, double []gobs, double []gcal, double []error, double []PARA, double FUNCT, String Area_fe) {

            obj = new Object[i_no_obs + 1][4];

            DecimalFormat df = new DecimalFormat("0.###");
            DecimalFormat d = new DecimalFormat("0.##");
            for(int K = 1; K <= i_no_obs; K++){
                obj[K][0] = "" + dis[K];
                obj[K][1] = "" + df.format(gobs[K]);
                obj[K][2] = "" + df.format(gcal[K]);
                obj[K][3] = "" + df.format(error[K]);
            }

            obj[0][0] = "ITERATION";
            obj[0][1] = "=" +" "+ite;

            INGREXP_TableView.val.setText("");
            INGREXP_TableView.val.append("ITERATION NUMBER ="+ite+"\n");
            INGREXP_TableView.val.appendText("\n");
            INGREXP_TableView.val.append("INTERPRETED PARAMETERS:-\n");
            INGREXP_TableView.val.appendText("-------------------------------------------\n");
            INGREXP_TableView.val.appendText("\n");
            INGREXP_TableView.val.appendText("DEPTH TO THE BOTTOM OF THE FAULT =" + d.format(PARA[1])+"(km)\n");
            INGREXP_TableView.val.appendText("\n");
            INGREXP_TableView.val.appendText("OBJECTIVE FUNCION  ="+df.format(FUNCT)+"\n");
            INGREXP_TableView.val.appendText("\n");

            INGREXP_TableView.val.appendText("\n");
            INGREXP_TableView.val.appendText("COEFFICIENTS OF THE POLYNOMIAL:-");
            INGREXP_TableView.val.appendText("\n");
            INGREXP_TableView.val.appendText("----------------------------------------------------\n");
            INGREXP_TableView.val.appendText(df.format(PARA[2])+"\n");
            INGREXP_TableView.val.appendText(df.format(PARA[3])+"\n");
            INGREXP_TableView.val.appendText(df.format(PARA[4])+"\n");

        }
        public static void drawGraph(){
            final com.ingrexp.view.INGREXP_DrawGraph dg = new com.ingrexp.view.INGREXP_DrawGraph();
            try
            {
                int width = 1280;
                int height = 650;
                BufferedImage buffer = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buffer.createGraphics();
                g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);
                Graphics2D g2 = (Graphics2D)g1 ;
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dg.drawGraph(g2);
                dg.drawOBJ(g2);
                dg.plotZCoordinates(g2);
                //dg.plotXYCoordinates(g2);
                dg.drawSd(g2);
                dg.plot(g2);
                dg.idex(g2);

                FileOutputStream os = new FileOutputStream( INGREXP_CalculateValues.input_area_name +".jpg");
                ImageIO.write(buffer, "jpg", os);
                os.close();

                String path =  INGREXP_CalculateValues.input_area_name  +".jpg";
```

```
                image = ImageIO.read(new File(path));

                Graphics g_image = INGREXP_MainPanel.img.getGraphics();
                g_image.drawImage(image, -60, -30, image.getWidth(), image.getHeight(), dg);
                MouseListener ml3 = new MouseAdapter(){
                    public void mouseClicked(MouseEvent e){
                        Graphics g_image = INGREXP_MainPanel.img.getGraphics();
                        g_image.drawImage(image, -60,-30,image.getWidth(), image.getHeight(),dg);
                    }
                };
                INGREXP_MainPanel.img.addMouseListener(ml3);
            }
            catch (Exception e2) {

                //  e2.printStackTrace();
            }
        }
    }
    -----------------------------------------------------------------------------------------
    package com.ingrexp.model;

    public class INGREXP_NOREQ {
        public static double []vsd = null;
        public static double []dep = null;
        public static int N2 = 0;

        public static void main(String[] args) {

            //Methods that support the main class
        }



        public static double []getGF(int n,int NNP,int deg,double ele[],double sd,double
    lambda,double []x,double par[],double []GC) {

            double Z1 = 0.00;
            double []Z ;
            double []GS ;
            double GGC = 0;
            double tsum;

            for (int K = 1; K <= n; K++ ) {
                GC[K] = 0;
            }
            double DX = ( x[2] - x[1] ) / 10;
            double ZB = par[1] - Z1;

            int ND = (int)( ZB / DX ) + 1;
            int N1 = ND / 2;
            if (ND - (2 * N1 ) < 0 || ND - ( 2 * N1 ) > 0) {
                ND = ND + 1;
            }
            double DZ = ZB / ND;
            N2 = ND + 1;
            Z = new double[N2 + 1];
            GS = new double[N2 + 1];
            vsd = new double[N2 + 1];
            dep = new double[N2 + 1];
            for (int JZ = 1; JZ <= N2; JZ++) {
                Z[JZ] = Z1 + DZ * ( JZ - 1 );
            }
            for (int K = 1; K <= n; K++) {
                double XX = x[K] ;
                for ( int JZ = 1;JZ <= N2; JZ++) {
                    double dc = sd * Math.exp(-lambda*Z[JZ]);
                    vsd[JZ] = dc;
                    dep[JZ] = Z[JZ];
                    tsum = 0;
                    for(int kk = 2; kk <= deg + 2; kk++){
                        tsum = tsum + par[kk] * Math.pow(Z[JZ],(kk - 2));
                    }

```

```
                    double tr1 = 3.14159265 / 2.0;
                    double tr2 = Math.atan( ( ( -XX + tsum) ) / ( ( Z[JZ] - ele[K] )  ) );
                    GS[JZ] = 13.3333 * dc * (tr1 - tr2);

                }
                GC[K] = getSIMP(GS, Z, N2, GGC);
            }
            return GC;
        }
        public static double getSIMP(double []gs,double []z,int n,double ggc) {

            double dz = z[2] - z[1];
            double sum1 = 0.0;
            double sum2 = 0.0;
            int n1 = n / 2;
            int n4 = n1 - 1;
            for(int I = 1; I <= n1; I++) {
                int n2 = 2 * I;
                sum1 = sum1 + gs[n2];
            }
            for(int I = 1; I <= n4; I++) {
                int n3 = 2 * I +1;
                sum2 = sum2 + gs[n3];
            }
            ggc = gs[1] + 4 * sum1 + 2 * sum2 + gs[n];
            ggc = ggc * dz / 3.0;
            return ggc;
        }
        public static double []getNOREQ(double p[][], double b[], int n, double KS[]) {

            int I = n + 1;
            double []a = new double[n * n + 1];
            for (int I1 = 1; I1 <= n; I1++) {
                for (int I2 = 1; I2 <= n; I2++) {

                    int I3 = (I1 - 1) * n + I2;
                    a[I3] = p[I2][I1];
                }
            }

            for (int I4 = 1;I4 <= n; I4++) {
                b[I4] = p[I4][I];
            }
            double TOL = 0;
            KS[0] = 0;
            int JJ = - n;
            int IT;
            int NY = 0;
            for (int J = 1;J <= n; J++) {
                int JY = J + 1;
                JJ = JJ + n + 1;
                double biga = 0;
                IT = JJ - J;
                int imax = 0;
                for (int i = J; i <= n; i++) {
                    int IJ = IT + i;
                    if (Math.abs(biga) - Math.abs(a[IJ]) < 0) {
                        biga = a[IJ];
                        imax = i;
                    }
                }
                int I1 = 0;
                if (Math.abs(biga) - TOL <= 0) {
                    KS[1] = 1;
                    return KS;
                }
                else {
                    I1 = J + n * (J - 2);
                    IT = imax - J;
                }
                double save;
                for (int K = J;K <= n; K++) {
                    I1 = I1 + n;
                    int I2 = I1 + IT;
                    save = a[I1];
```

```
                    a[I1] = a[I2];
                    a[I2] = save;
                    a[I1] = a[I1] / biga;
                }
                save = b[imax];
                b[imax] = b[J];
                b[J] = save / biga;
                int IQS = 0;
                if (J - n < 0 || J - n > 0) {
                    IQS = n * (J - 1);
                    for (int IX = JY;IX <= n; IX++) {
                        int IXJ = IQS + IX;
                        IT = J - IX;
                        for (int JX = JY;JX <= n; JX++) {
                            int IXJX = n * (JX - 1) + IX;
                            int JJX = IXJX + IT;
                            a[IXJX] = a[IXJX] - (a[IXJ] * a[JJX]);
                        }
                        b[IX] = b[IX] - (b[J] * a[IXJ]);
                    }
                }
            }
            NY = n - 1;
            IT = n * n;
            for (int J = 1;J <= NY; J++) {
                int ia = IT - J;
                int ib = n - J;
                int ic = n;
                for (int K = 1;K <= J; K++) {
                    b[ib] = b[ib] - a[ia] * b[ic];
                    ia = ia - n;
                    ic = ic - 1;
                }
            }
            return b;
        }
    }
    ---------------------------------------------------------------------------------------------

    package com.ingrexp.control;

    import java.awt.Color;
    import java.awt.Graphics;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.io.File;
    import java.io.FileWriter;
    import java.io.IOException;
    import java.text.DecimalFormat;

    import javax.swing.JFileChooser;
    import javax.swing.JFrame;
    import javax.swing.JOptionPane;

    import com.ingrexp.model.INGREXP_CalculateValues;
    import com.ingrexp.view.INGREXP_MainPanel;
    import com.ingrexp.view.INGREXP_TableView;

    public class INGREXP_Controller implements ActionListener{
        String rowdata[][]={};
        com.ingrexp.model.INGREXP_CalculateValues cv = new
    com.ingrexp.model.INGREXP_CalculateValues();
        FileWriter myWriter = null;
        public static boolean success = false;

        public void actionPerformed(ActionEvent ae) {

            if(ae.getActionCommand().equals("Interpretation")) {

                com.ingrexp.view.INGREXP_TableView.populateEastPanel(rowdata);
                INGREXP_TableView.val.setText("");
                cv.getAnamolyValues(com.ingrexp.view.INGREXP_MainPanel.captureValues());
                com.ingrexp.view.INGREXP_TableView.populateEastPanel(INGREXP_CalculateValues.ob
    );
```

```java
                INGREXP_CalculateValues.obj = null;
                com.ingrexp.view.INGREXP_MainPanel.p_East.repaint();
                com.ingrexp.view.INGREXP_MainView mv = new com.ingrexp.view.INGREXP_MainView();
                mv.setResizable(true);

            } else if(ae.getActionCommand().equals("Save & Print")){

                try{
                    String current = System.getProperty("user.dir");
                    File img_file = new File( INGREXP_CalculateValues.input_area_name+".jpg");
                    JFileChooser saveFile = new JFileChooser(current);
                    File OutFile = saveFile.getSelectedFile();
                    if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
                    {
                        OutFile = saveFile.getSelectedFile();

                        if (OutFile.canWrite() || !OutFile.exists())
                        {
                            File dir = new File(OutFile.getParent());
                            success = img_file.renameTo(new File(dir,img_file.getName()));
                            System.out.println("save successful" + success);
                            myWriter = new FileWriter(OutFile+".html");
                            myWriter.write(" </table> </td> <td> <img src = '"+ INGREXP_CalculateValues.input_area_name +".jpg'></td></tr></table>");
                            myWriter.write("<html><Body onLoad = \"window.print()\"> <table> <tr> <td>" +
                                            "<table border = 1> <tr> <th colspan = 4>LOCATION:-"+INGREXP_CalculateValues.input_area_name+"</th> </tr>");



                            DecimalFormat df =new DecimalFormat("0.###");
                            myWriter.write(" <tr><th colspan = 4> PROFILE NUMBER:-"+" "+INGREXP_CalculateValues.input_profile_num+" </th></tr>");
                            myWriter.write(" <tr><th colspan = 4> ITERATION"+" "+INGREXP_CalculateValues.o_iter+" </th></tr>");

                            myWriter.write("<tr > <th>Distance (km) </th> <th> Observed anomalies (mGal) </th> <th> Calculated anomalies (mGal) </th> <th> Error (mGal) </th></tr>");

                            for ( int K = 1; K <= INGREXP_CalculateValues.input_n_obs; K++){

                                myWriter.write("<tr> <td>" + INGREXP_CalculateValues.input_x_km[K]+"</td> <td>"+df.format(INGREXP_CalculateValues.input_nob_gob[K])+"</td> <td>"+df.format(INGREXP_CalculateValues.o_GC[K])+"</td> <td>"+df.format(INGREXP_CalculateValues.o_err[K])+"</td></tr>");
                            }
                            //  myWriter.write(" </table> </td> <td> <img src = '"+ INGREXP_CalculateValues.input_area_name+INGREXP_CalculateValues.input_profile_num +".jpg'></td></tr></table><BR> Interpreted Parameters :<BR>");
                            myWriter.write("</table>");
                            myWriter.write("INTERPRETED PARAMETERS: <BR>");
                            myWriter.write("------------------------------------ <BR>");
                            DecimalFormat d =new DecimalFormat("0.##");
                            DecimalFormat d1 =new DecimalFormat("0.###");

                            myWriter.write("DEPTH TO THE BOTTOM OF THE FAULT  ="+" "+d.format(INGREXP_CalculateValues.o_par[1])+"(km) <BR>");
                            myWriter.write("OBJECTIVE FUNCTION ="+" "+d1.format(INGREXP_CalculateValues.o_func)+"<BR>");
                            myWriter.write("<BR>");

                            for ( int i = 2; i <= 4; i++) {
                                if ( i == 2 ) {
                                    myWriter.write("COEFFICIENTS OF THE POLYNOMIAL:-"+"<BR>");
                                }
                                myWriter.write(d1.format(INGREXP_CalculateValues.o_par[i])+"<BR>");
                            }
                            myWriter.close();
                        }
```

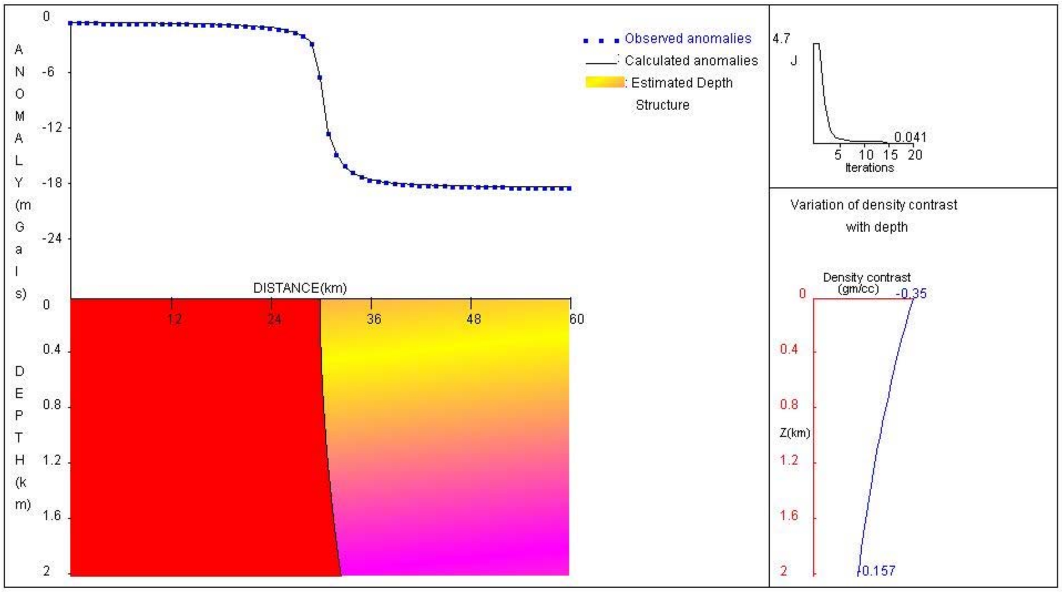
```
                    }
                    else
                    {
                        //pops up error message
                    }
                }
                catch(Exception e1) {

                    e1.printStackTrace();
                }

            }else if(ae.getActionCommand().equals("Load data")){
                try {
                    INGREXP_MainPanel.loadData1();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }else if(ae.getActionCommand().equals("Clear")){

                INGREXP_MainPanel.clearDefaultValues();
                com.ingrexp.view.INGREXP_MainPanel.clearPanel(INGREXP_MainPanel.p_Center);
                com.ingrexp.view.INGREXP_TableView.populateEastPanel(rowdata);
                INGREXP_TableView.val.setText("");
                Graphics g = INGREXP_MainPanel.img.getGraphics();
                g.setColor(Color.white);
                g.fillRect(0, 0, 1000, 600);

            }else if(ae.getActionCommand().equals("Exit")){
                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit INGREXP ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(success==false){
                        String fileName = INGREXP_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        }
}
---------------------------------------------------------------------------------------------

package com.ingrexp.util;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

import com.ingrexp.model.INGREXP_CalculateValues;

public class INGREXP_Utility {
    public static double convertDouble(String str) throws Exception {


        Double temp = null;

        try {
            temp = new Double(str.trim());

        }
        catch(Exception e){
            JFrame frame = null;
            JOptionPane.showMessageDialog(frame,
                    "Enter a numerical value.",
                    "Number format error",
                    JOptionPane.ERROR_MESSAGE);
        }
        return temp.doubleValue();
    }

```

```java
    public static String convertString(String str) throws Exception {

        String temp = new String(str.trim());
        return temp;
    }

    public static int convertInteger(String str) throws Exception {
        Integer temp = null;
        try {
            temp = new Integer(str.trim());
        }
        catch(Exception e){

            JFrame frame = null;
            JOptionPane.showMessageDialog(frame,
                    "Enter a numerical value.",
                    "Number format error",
                    JOptionPane.ERROR_MESSAGE);

        }
        return temp.intValue();
    }

    public static double findMaximumNumber( double observe[]) {

        double max = 0.0d;
        for (int i = 0; i < observe.length; i++) {

            if (Math.abs(observe[i]) > Math.abs(max)) {

                max = observe[i];
            }
        }

        double maxVal =  max/3*5;
        return maxVal;
    }

    public static double findMinimumNumber( double observe[], double denVal) {

        double max = denVal;
        for (int i = 1; i < observe.length; i++) {

            if (Math.abs(observe[i]) < Math.abs(max)) {

                max = Math.abs(observe[i]);
            }
        }

        double maxVal =  max;
        return maxVal;
    }

    public static double findMinimumNumber1( double observe[]) {

        double max = 0.0d;
        for (int i = 1; i < observe.length; i++) {

            if ((observe[i]) < (max)) {

                max = (observe[i]);
            }
        }

        double maxVal =  max;
        return maxVal;
    }
    public static double findMaximumNumber1( double observe[]) {

        double max = 0.0d;
        for (int i = 1; i < observe.length; i++) {

            if (Math.abs(observe[i]) > Math.abs(max)) {

```

```
                max =Math.abs(observe[i]);
            }
        }

        double maxVal =  max;
        return maxVal;
    }
    public static double findMaximumNumber( double observe[], double anoVal) {

        double max = anoVal;
        for (int i = 1; i < observe.length; i++) {

            if ((observe[i]) > (max)) {

                max = (observe[i]);
            }
        }

        double maxVal =  max;
        return maxVal;
    }
    public static int findMaximumNumber( double observe) {

        double max = 0.0d;
        int maxVal=0;
        max = observe;

        if (max < 5) {

            maxVal = 5;
        }
        else if (max >= 5 && max <= 10) {

            maxVal = 10;
        }
        else if ( max > 10 && max <= 15) {

            maxVal = 15;
        }
        else if (max > 15 && max <= 20) {

            maxVal = 20;
        }
        else
        {
            maxVal = INGREXP_CalculateValues.o_iter;
        }
        return maxVal;
    }


    public static double[] convertDoubleArray(String str) throws Exception {

        java.util.StringTokenizer st = new java.util.StringTokenizer(str, ",");
        String temp = "";
        java.util.ArrayList arr = new java.util.ArrayList();

        while(st.hasMoreTokens()) {

            temp = st.nextToken();
            arr.add(temp);
        }
        double d_array[] = new double[arr.size() + 1];

        for (int i = 0; i <= arr.size(); i++) {

            if (i == 0)
                d_array[i] = 0.0;
            else {

                try {
                    d_array[i] = convertDouble( arr.get(i - 1).toString() );
                }
                catch(Exception e){
                    JFrame frame = null;
```

```
                        JOptionPane.showMessageDialog(frame,
                                "Enter numerical values.",
                                "Number format error",
                                JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
            return d_array;
        }
    }
```

Annexure - 2B
Sample output



LOCATION:- sample			
PROFILE NUMBER:- 1			
ITERATION 14			
Distance (km)	**Observed anomalies (mGal)**	**Calculated anomalies (mGal)**	**Error (mGal)**
0.0	-0.178	-0.179	0.001
1.0	-0.184	-0.184	0
2.0	-0.191	-0.191	-0
3.0	-0.198	-0.198	-0
4.0	-0.205	-0.205	-0
5.0	-0.212	-0.213	0.001
6.0	-0.221	-0.221	0
7.0	-0.23	-0.23	0
8.0	-0.24	-0.24	0
9.0	-0.251	-0.251	-0
10.0	-0.263	-0.263	-0
11.0	-0.276	-0.276	-0
12.0	-0.29	-0.29	0
13.0	-0.306	-0.306	0
14.0	-0.324	-0.324	-0
15.0	-0.344	-0.344	-0
16.0	-0.366	-0.367	0.001
17.0	-0.392	-0.392	0
18.0	-0.422	-0.422	0
19.0	-0.456	-0.457	0.001
20.0	-0.497	-0.497	0
21.0	-0.546	-0.546	0
22.0	-0.605	-0.606	0.001
23.0	-0.679	-0.68	0.001
24.0	-0.774	-0.774	0
25.0	-0.9	-0.9	-0
26.0	-1.076	-1.074	-0.002
27.0	-1.339	-1.333	-0.006
28.0	-1.777	-1.762	-0.015
29.0	-2.672	-2.63	-0.042
30.0	-6.614	-6.817	0.203
31.0	-13.512	-13.335	-0.177
32.0	-15.998	-16.01	0.012
33.0	-17.367	-17.474	0.107
34.0	-18.184	-18.278	0.094
35.0	-18.682	-18.739	0.057
36.0	-18.995	-19.026	0.031
37.0	-19.204	-19.219	0.015
38.0	-19.351	-19.357	0.006
39.0	-19.459	-19.461	0.002

40.0	-19.543	-19.541	-0.002
41.0	-19.609	-19.605	-0.004
42.0	-19.663	-19.657	-0.006
43.0	-19.707	-19.701	-0.006
44.0	-19.745	-19.737	-0.008
45.0	-19.777	-19.769	-0.008
46.0	-19.804	-19.796	-0.008
47.0	-19.828	-19.82	-0.008
48.0	-19.85	-19.841	-0.009
49.0	-19.869	-19.859	-0.01
50.0	-19.885	-19.876	-0.009
51.0	-19.9	-19.891	-0.009
52.0	-19.914	-19.904	-0.01
53.0	-19.926	-19.917	-0.009
54.0	-19.938	-19.928	-0.01
55.0	-19.948	-19.938	-0.01
56.0	-19.958	-19.948	-0.01
57.0	-19.966	-19.956	-0.01
58.0	-19.974	-19.964	-0.01
59.0	-19.982	-19.972	-0.01
60.0	-19.989	-19.979	-0.01

INTERPRETED PARAMETERS:

DEPTH TO THE BOTTOM OF THE FAULT = 2(km)
OBJECTIVE FUNCTION = 0.041

COEFFICIENTS OF THE POLYNOMIAL:-
30
0.184
0.544

Annexure - 3A
FRGMLSTRK

```
package com.frgmlstrk.view;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

import com.frgmlstrk.control.FRGMLSTRK_Controller;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;

public class FRGMLSTRK_MainView extends Frame{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static void main(String s[])
    {
        FRGMLSTRK_MainView cm = new FRGMLSTRK_MainView();
        cm.setSize(1280, 768);
        cm.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit FRGMLSTRK ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(FRGMLSTRK_Controller.success == false){
                        String fileName = FRGMLSTRK_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        });
        cm.setTitle("FRGMLSTRK");
        cm.setResizable(true);
        cm.add(new FRGMLSTRK_MainPanel(cm));
        cm.setVisible(true);

    }

}
------------------------------------------------------------------------------------------

package com.frgmlstrk.view;

import java.awt.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.HashMap;
import javax.swing.JFileChooser;
import com.frgmlstrk.control.FRGMLSTRK_Controller;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotDensity;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotDepth;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotFault;

public class FRGMLSTRK_MainPanel extends Panel {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static TextArea img = new TextArea(46,140);

```

```
        public static TextArea img1 = new TextArea(46,140);
        public static TextArea img2 = new TextArea(46,140);
        public static TextArea img3 = new TextArea(46,140);
        public static TextArea img4 = new TextArea(46,140);
        public static TextArea fl = new TextArea(46,140);
        public static TextArea den = new TextArea(46,140);
        public static TextArea im = new TextArea(46,140);

        Panel p_North, p_West,p_South;
        public static Panel p_East;
        public static Label graphLabel;
        public static Panel p_Center;
        public static TextField inputValues [] = new TextField[18];

        Button actionButton[] = new Button[12];
        Object rowdata[][]={};

        /**Field Area Name*/
        public final static int AREA_FE = 0;
        /**Number of the Profile*/
        public final static int NUM_PROFILE =   1;
        /**Strike of the structure(km)*/
        public static final int STR_ST=2 ;
        /**Number of density interfaces*/
        public static final int NOB_DI=3;
        /**Maximum depth(km)*/
        public static final int MAX_DEP=4;
        /**Basement density(gm/cc)*/
        public static final int BASE_DEN=5;
        /**Number of observation*/
        public static final int N_OBS = 6 ;
        /**Offset of the profile(km)*/
        public static final int OFF_PRO=7;
        /**Distances(km)*/
        public static final int DIS_KM=8;
        /**Elevation(km)*/
        public static final int ELE_KM=9;
        /**Observed anomaly*/
        public static final int OBS_ANO=10;
        /** Degree of polynomial*/
        public static final int D_POLY=11;
        /** Maximum Density*/
        public static final int MAX_DEN=12;
        /** Minimum Density*/
        public static final int MIN_DEN=13;

        public FRGMLSTRK_MainPanel(FRGMLSTRK_MainView cm){

            this.setLayout(new BorderLayout());
            p_North = new Panel();
            p_West = new Panel();
            p_East = new Panel ();
            p_South = new Panel();
            p_Center = new Panel();

            graphLabel = new Label("Interactive Gravity Modeling of Strike Limited Listric Fau
    Sources", Label.CENTER);
            graphLabel.setFont(new Font("Arial", 40, 20));
            p_Center.add(graphLabel);

            for(int i = 0; i < 15; i++){
                inputValues[i] = new TextField();
            }

            actionButton[0] = new Button("Specify fault coordinates");
            actionButton[1] = new Button("Draw/Edit fault plane");
            actionButton[2] = new Button("Specify depth interfaces");
            actionButton[3] = new Button("Specify density values");
            actionButton[4] = new Button("Forward Modeling");
            actionButton[5] = new Button("Graph");
            actionButton[6] = new Button("Save and Print");
            actionButton[7] = new Button("Sample data");
            actionButton[8] = new Button("Clear");
            actionButton[9] = new Button("Save file");
            actionButton[10] = new Button("Load file");
```

```
            actionButton[11] = new Button("Exit");

            this.populateNorthPanel();
            FRGMLSTRK_TableView.populateEastPanel(rowdata);

            this.add(p_North, BorderLayout.NORTH);
            p_Center.setSize(1000, 760);
            this.add(p_Center, BorderLayout.CENTER);
            p_Center.add(img);
            this.add(p_East, BorderLayout.EAST);
            MenuBar mb = new MenuBar();
            cm.setMenuBar(mb);
            Menu file = new Menu("File");
            Menu file2 = new Menu("Fault");
            Menu file3 = new Menu("Depth");
            Menu file4 = new Menu("Interpretation");
            MenuItem i1,i2,i3,i5,i6,i7,i8,i9,i10,i11,i12,i13;
            file.add(i1 = new MenuItem("New"));
            file.add(i2 = new MenuItem("Load file"));
            file.add(i3 = new MenuItem("Save file"));
            file.add(i5 = new MenuItem("Clear"));
            file.add(i6 = new MenuItem("Exit"));

            file2.add(i7 = new MenuItem("Specify fault coordinates"));
            file2.add(i8 = new MenuItem("Draw/Edit fault plane"));

            file3.add(i9 = new MenuItem("Specify depth interfaces"));
            file3.add(i10 = new MenuItem("Specify density values"));

            file4.add(i11 = new MenuItem("Forward Modeling"));
            file4.add(i12 = new MenuItem("Graph"));
            file4.add(i13 = new MenuItem("Save and Print"));
            mb.add(file);
            mb.add(file2);
            mb.add(file3);
            mb.add(file4);
            i1.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i2.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i3.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i5.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i6.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i7.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i8.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i9.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i10.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i11.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i12.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());
            i13.addActionListener(new com.frgmlstrk.control.FRGMLSTRK_Controller());




            this.setVisible(true);
        }

        public void populateNorthPanel(){
            p_North.setLayout(new GridLayout(5,6));

            p_North.add(new Label("Area Name"));
            p_North.add(inputValues[0]);

            p_North.add(new Label("Profile Name"));
            p_North.add(inputValues[1]);

            p_North.add(new Label("Half strike length (km)"));
            p_North.add(inputValues[2]);

            p_North.add(new Label("Number of density interfaces"));
            p_North.add(inputValues[3]);

            p_North.add(new Label("Depth to the basement(km)"));
            p_North.add(inputValues[4]);

            p_North.add(new Label("Basement density(gm/cc)"));
            p_North.add(inputValues[5]);
```

```java

            p_North.add(new Label("Number of observations"));
            p_North.add(inputValues[6]);

            p_North.add(new Label("Offset of the profile(km)"));
            p_North.add(inputValues[7]);

            p_North.add(new Label("Distance(km)"));
            p_North.add(inputValues[8]);

            p_North.add(new Label("Elevation(km)"));
            p_North.add(inputValues[9]);

            p_North.add(new Label("Observed anomaly"));
            p_North.add(inputValues[10]);

            p_North.add(new Label("Degree of polynomial"));
            p_North.add(inputValues[11]);

            p_North.add(new Label("Maximum density"));
            p_North.add(inputValues[12]);

            p_North.add(new Label("Minimum density"));
            p_North.add(inputValues[13]);

            p_North.add(actionButton[0]);
            p_North.add(actionButton[1]);
            p_North.add(actionButton[2]);
            p_North.add(actionButton[3]);
            p_North.add(actionButton[4]);
            p_North.add(actionButton[5]);
            p_North.add(actionButton[6]);
            p_North.add(actionButton[7]);
            p_North.add(actionButton[8]);
            p_North.add(actionButton[9]);
            p_North.add(actionButton[10]);
            p_North.add(actionButton[11]);

            actionButton[0].addActionListener(new FRGMLSTRK_Controller());
            actionButton[1].addActionListener(new FRGMLSTRK_Controller());
            actionButton[2].addActionListener(new FRGMLSTRK_Controller());
            actionButton[3].addActionListener(new FRGMLSTRK_Controller());
            actionButton[4].addActionListener(new FRGMLSTRK_Controller());
            actionButton[5].addActionListener(new FRGMLSTRK_Controller());
            actionButton[6].addActionListener(new FRGMLSTRK_Controller());
            actionButton[7].addActionListener(new FRGMLSTRK_Controller());
            actionButton[8].addActionListener(new FRGMLSTRK_Controller());
            actionButton[9].addActionListener(new FRGMLSTRK_Controller());
            actionButton[10].addActionListener(new FRGMLSTRK_Controller());
            actionButton[11].addActionListener(new FRGMLSTRK_Controller());
        }


        public static HashMap captureValues(){

            HashMap h_Map = new HashMap();
            try {

                h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
                h_Map.put("NUM_PROFILE", inputValues[NUM_PROFILE].getText());
                h_Map.put("N_OBS", inputValues[N_OBS].getText());
                h_Map.put("D_POLY", inputValues[D_POLY].getText());
                h_Map.put("STR_ST", inputValues[STR_ST].getText());
                h_Map.put("OFF_PRO", inputValues[OFF_PRO].getText());
                h_Map.put("DIS_KM", inputValues[DIS_KM].getText());
                h_Map.put("ELE_KM",inputValues[ELE_KM].getText());
                h_Map.put("NOB_DI", inputValues[NOB_DI].getText());
                h_Map.put("MAX_DEP", inputValues[MAX_DEP].getText());
                h_Map.put("BASE_DEN", inputValues[BASE_DEN].getText());
                h_Map.put("OBS_ANO", inputValues[OBS_ANO].getText());
                h_Map.put("MAX_DEN", inputValues[MAX_DEN].getText());
                h_Map.put("MIN_DEN", inputValues[MIN_DEN].getText());

            }
            catch (Exception e) {
```

```java
                e.printStackTrace();
            }

            return h_Map;

        }

        public static void clearPanel(TextArea p) {

            Graphics g = p.getGraphics();
            g.setColor(Color.WHITE);
            g.fillRect(0, 0, 1000, 600);
        }



        public static void setDefaultValues(){

            inputValues[AREA_FE].setText("SAMPLE");
            inputValues[NUM_PROFILE].setText("1");
            inputValues[N_OBS].setText("50");
            inputValues[D_POLY].setText("9");
            inputValues[STR_ST].setText("40");
            inputValues[OFF_PRO].setText("30");
            inputValues[DIS_KM].setText("1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
2,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,5
0");
            inputValues[ELE_KM].setText("0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0");
            inputValues[NOB_DI].setText("3");
            inputValues[MAX_DEP].setText("8");
            inputValues[BASE_DEN].setText("2.67");
            inputValues[OBS_ANO].setText("-0.388,-0.403,-0.418,-0.435,-0.452,-0.471,-0.491,-0.5
2,-0.534,-0.558,-0.585,-0.613,-0.644,-0.68,-0.721,-0.772,-0.838,-0.933,-1.092,-1.408
,-2.228,-7.712,-14.508,-16.295,-16.323,-15.878,-15.393,-15.025,-14.838,-14.857,-15.0
81,-15.49,-16.049,-16.719,-17.459,-18.234,-19.018,-19.791,-20.539,-21.253,-21.929,-2
2.564,-23.159,-23.712,-24.225,-24.699,-25.134,-25.53,-25.888,-26.209");
            inputValues[MAX_DEN].setText("3");
            inputValues[MIN_DEN].setText("2");
        }

        public static void loadData(){
            try{

                JFileChooser chooser=new  JFileChooser();
                int returnVal = chooser.showOpenDialog(null);
                int count = 0;
                if(returnVal == JFileChooser.APPROVE_OPTION) {
                    File f = chooser.getSelectedFile();
                    BufferedReader br=new BufferedReader(new FileReader(f));
                    String st;
                    st = br.readLine();
                    count++;

                    while((st)!=null){

                        try {

                            if (count==1){

                                FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.AREA_FE].se
Text(""+st);

                            }
                            if (count==2){

                                FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.NUM_PROFILE
.setText(""+st);

                            }
                            if (count==3){

                                FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.STR_ST].set
ext(""+st);

```

```
                    }
                    if (count==4){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.NOB_DI].set
ext(""+st);
                        FRGMLSTRK_CalculateValues.inter =
FRGMLSTRK_Utility.convertInteger(st)+2;
                    }
                    if (count==5){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.MAX_DEP].se
Text(""+st);

                    }
                    if (count==6){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.BASE_DEN].s
tText(""+st);

                    }
                    if (count==7){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.N_OBS].setT
xt(""+st);

                    }
                    if (count==8){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.OFF_PRO].se
Text(""+st);

                    }
                    if (count==9){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.DIS_KM].set
ext(""+st);

                    }
                    if (count==10){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.ELE_KM].set
ext(""+st);

                    }
                    if (count==11){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.OBS_ANO].se
Text(""+st);

                    }
                    if (count==12){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.D_POLY].set
ext(""+st);

                    }
                    if (count==13){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.MAX_DEN].se
Text(""+st);

                    }
                    if (count==14){

                        FRGMLSTRK_MainPanel.inputValues[FRGMLSTRK_MainPanel.MIN_DEN].se
Text(""+st);

                    }

                    if (count==15){
                        FRGMLSTRK_PlotFault.val =
FRGMLSTRK_Utility.convertDoubleArray(st);

                    }
                    if (count==16){
```

```
                                   FRGMLSTRK_PlotFault.val1 =
FRGMLSTRK_Utility.convertDoubleArray(st);

                               }
                               if (count==17){
                                   FRGMLSTRK_CalculateValues.len =
FRGMLSTRK_Utility.convertInteger(st);

                               }
                               if (count==18){
                                   FRGMLSTRK_PlotDepth.val2 =
FRGMLSTRK_Utility.convertDoubleArray(st);

                               }
                               if (count==19){
                                   FRGMLSTRK_PlotDensity.val3 =
FRGMLSTRK_Utility.convertDoubleArray(st);

                               }
                               if (count==20){
                                   FRGMLSTRK_CalculateValues.inter =
FRGMLSTRK_Utility.convertInteger(st);

                               }
                           }
                           catch(Exception e) {

                               e.printStackTrace();
                           }
                           st = br.readLine();
                           count++;

                       }
                   }
               }
               catch(Exception e){

               }
           }

           public static void clearDefaultValues(){

               inputValues[AREA_FE].setText("");
               inputValues[NUM_PROFILE].setText("");
               inputValues[N_OBS].setText("");
               inputValues[D_POLY].setText("");
               inputValues[STR_ST].setText("");
               inputValues[OFF_PRO].setText("");
               inputValues[DIS_KM].setText("");
               inputValues[ELE_KM].setText("");
               inputValues[NOB_DI].setText("");
               inputValues[MAX_DEP].setText("");
               inputValues[BASE_DEN].setText("");
               inputValues[OBS_ANO].setText("");
               inputValues[MAX_DEN].setText("");
               inputValues[MIN_DEN].setText("");

           }
       }
       ---------------------------------------------------------------------------------------------

       package com.frgmlstrk.view;

       import java.awt.*;
       import javax.swing.JScrollPane;
       import javax.swing.JTable;


       public class FRGMLSTRK_TableView extends Panel{

           /**
            *
            */
           private static final long serialVersionUID = 1L;

```

```java
    public static TextArea val = new TextArea(5,5);
    public static  void populateEastPanel(Object rowData[][]) {

        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.removeAll();
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.setLayout(new GridLayout(2,1));

        Object columnNames[] = {"Distance(km)","Observed anomalies (mGal)" ,"Calculated anomalies (mGal)"};
        JTable table = new JTable(rowData, columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(300,500));
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setAutoscrolls(true);
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.add(scrollPane);
        val.setEditable(false);
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.add(val);
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.validate();
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.p_East.setVisible(true);
    }
}
------------------------------------------------------------------------------------------

package com.frgmlstrk.view;


import java.awt.*;
import java.applet.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.graph.FRGMLSTRK_DensityGraph;


public class FRGMLSTRK_DrawGraph extends Applet {


    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static int i_no_obs;
    float maxY,maxZ,maxX ;
    public static float strike,strike1,strike2;
    double obs[];

    public void drawGraph(Graphics2D g2) {

        g2.setFont(new Font("Arial", 20, 12));
        g2.setColor(Color.BLACK);
        g2.draw(new Line2D.Float(215-strike1, 50, 215-strike1, 300));
        String []a = {"A", "N", "O", "M", "A", "L", "Y", "(m", "G", "a", "l", "s)"};
        String []b = {"D", "E", "P", "T", "H", "(k", "m)"};

        for (int i = 0; i < a.length; i++) {

            g2.drawString(""+a[i], 100, 20 + 60 + ( i * 20 ) );
        }
        for (int i = 0; i < b.length; i++) {

            g2.drawString(""+b[i], 100, 20 + 350 + ( i * 20 ) );
        }

    }

    public void plot(Graphics2D g2) {

        g2.setFont( new Font("Arial", 12, 12) );
        DecimalFormat f = new DecimalFormat("0.#");
        g2.setColor(Color.BLACK);
        strike = (float) (FRGMLSTRK_CalculateValues.d_str_st-FRGMLSTRK_CalculateValues.d_off_pro);
        strike1 = (float) (65 * (strike / (2 * FRGMLSTRK_CalculateValues.d_str_st)));
        strike2= (float) (40 * (strike /(2 * FRGMLSTRK_CalculateValues.d_str_st)));

```

```
        i_no_obs = FRGMLSTRK_CalculateValues.i_no_obs;

        obs = new double[i_no_obs+1];
        for (int i = 1; i <= i_no_obs; i++) {

            obs[i] = FRGMLSTRK_CalculateValues.d_dis_km[i];
        }

        maxX = (float)obs[i_no_obs];
        maxY = FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.input_nob_gob)
        maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;

        g2.drawString("|",(float) 600 + 65 - strike1, 260 + strike2);
        g2.drawString(""+f.format(FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs]),(float) 600
+ 65 - strike1, 260 + strike2 - 8);
        g2.drawString("0", 125, 310);
        g2.drawString("DISTANCE(km)", 400, 250);
        float xplot = 0;
        float xInterval = (float) (FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs] / 5);

        int zInterval = 50;
        for (float x = xInterval, j = 1; x < 600; x += xInterval){

            xplot = xplot + xInterval;
            if(j > 4)
                break;
            g2.drawString("|",(float) (215 + (450 * x / maxX) - strike1), 260 + strike2);
            g2.drawString("" + f.format(xplot), (float) (215 + (450 * x / maxX) - strike1)
3, 260 + strike2 - 8);
            j++;
        }

        DecimalFormat d =  new DecimalFormat("0.#");
        float points1 = maxZ / 5 ;
        for (int x = zInterval + 250,j = 1; x < 550; x += zInterval){

            g2.drawString("-", 148, 52 + x);
            g2.drawString("" +d.format(points1 * j), 125, 50 + x);
            j++;
        }

    }


    public void plotXYCoordinates(Graphics2D g2){

        double minAno =
FRGMLSTRK_Utility.findMinimumNumber1(FRGMLSTRK_CalculateValues.input_nob_gob);
        double maxAno =
FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.input_nob_gob,
minAno);
        double minObAno =
FRGMLSTRK_Utility.findMinimumNumber1(FRGMLSTRK_CalculateValues.ano);
        double maxObAno = FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.ano
minObAno);

        if (minAno < 0 && maxObAno < 0 && maxAno < 0 && minObAno <0 ){
            plotXYCoordinates1(g2);
        }
        if (minAno >= 0 && maxObAno > 0){
            plotXYCoordinates1(g2);
        }

        if (minAno < 0 && maxObAno > 0 || maxAno>0 && minObAno<0){
            plotXYCoordinates2(g2);
        }
    }

    public void plotXYCoordinates1 (Graphics2D g2) {

        g2.setFont(new Font("Arial", 20, 12));
        g2.setColor(Color.black);
        float maxval = (float)
FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.ano);
        float maxval1 = (float)
```

```
    FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.input_nob_gob);
            if(Math.abs(maxval)>Math.abs(maxval1))
                maxY = maxval;
            else
                maxY = maxval1;
            int points = (int)maxY / 5;
            int yInterval = 50;
            g2.drawString("0", 215 - strike1 - 40, 50);
            for (int x = yInterval, j = 1; x < 250; x += yInterval){

                g2.drawString("-", 215 - strike1 - 2, 50 + x);
                g2.drawString("" + (points*j), 215 - strike1 - 40, 50 + x);
                j++;
            }
            float prevx = 215 - strike1;
            float prevy = (float)( 50 + ( 250 * FRGMLSTRK_CalculateValues.ano[1] / maxY ) );
            float xpoint = 0;
            float ypoint = 0;
            float gypoint = 0;

            for (int k = 1; k <= i_no_obs; k++) {

                xpoint = (float)( 450 * obs[k] / maxX);
                ypoint = (float)( ( 250 * FRGMLSTRK_CalculateValues.ano[k] / maxY ) );
                gypoint = (float)( ( 250 * FRGMLSTRK_CalculateValues.input_nob_gob[k] / maxY )
    );

                g2.setColor(Color.BLACK);
                g2.draw(new Line2D.Float(prevx, prevy, 215-strike1+ xpoint, 50 + ypoint ));

                g2.setColor(Color.BLUE);
                g2.setFont(new Font("Arial", 20, 40));
                g2.drawString(".",  215-strike1+xpoint - 6 , 50 + gypoint + 3);

                g2.setFont(new Font("Arial", 20, 12));
                g2.setColor(Color.black);
                prevx = 215-strike1 + xpoint;
                prevy = 50 + ypoint ;

            }

        }


        public void plotXYCoordinates2 (Graphics2D g2) {


            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.black);

            double store[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
            double store1[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
            double negstore[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
            double negstore1[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
            for(int i = 1; i <= FRGMLSTRK_CalculateValues.i_no_obs; i++){
                if(FRGMLSTRK_CalculateValues.ano[i]>0)
                    store[i] = FRGMLSTRK_CalculateValues.ano[i];
                else
                    negstore[i] = FRGMLSTRK_CalculateValues.ano[i];
                if(FRGMLSTRK_CalculateValues.input_nob_gob[i]>0)
                    store1[i] = FRGMLSTRK_CalculateValues.input_nob_gob[i];
                else
                    negstore1[i] = FRGMLSTRK_CalculateValues.input_nob_gob[i];
            }
            float maxpos = (float) FRGMLSTRK_Utility.findMaximumNumber1(store);
            float maxpos1 = (float) FRGMLSTRK_Utility.findMaximumNumber1(store1);
            float maxneg = (float) FRGMLSTRK_Utility.findMaximumNumber(negstore);
            float maxneg1 = (float) FRGMLSTRK_Utility.findMaximumNumber(negstore1);
            float posnum =0;

            if(maxpos>maxpos1)
                posnum = maxpos;
            else
                posnum = maxpos1;
            if(Math.abs(maxneg)>Math.abs(maxneg1))
```

```
                maxY = maxneg;
            else
                maxY = maxneg1;

            float prevx = 215-strike1;
            float prevy = 0;
            if(FRGMLSTRK_CalculateValues.ano[1]>0)
                prevy = 100-(float)( ( 50 * FRGMLSTRK_CalculateValues.ano[1] / posnum ) );
            else
                prevy = 100+(float)( ( 200 * FRGMLSTRK_CalculateValues.ano[1] / maxY ) );

            float xpoint = 0;
            float ypoint = 0;
            float gypoint = 0;

            DecimalFormat df = new DecimalFormat("0.##");
            g2.drawString("0",215 - strike1 - 40,100);
            g2.drawString("-", 215-strike1, 55);
            g2.drawString(""+df.format(posnum), 215-strike1-40, 50);
            DecimalFormat f = new DecimalFormat("0.#");
            float points = maxY / 4;
            int yInterval=50;
            for (int x = yInterval, j = 1; x < 250; x+=yInterval){

                g2.drawString("-", 215 - strike1 - 2, 100 + x );
                g2.drawString("" + f.format(points * j), 215 - strike1 - 40, 100 + x );
                j++;
            }
            for (int k = 1; k <= i_no_obs; k++) {

                xpoint = (float)( 450 * obs[k] / maxX);
                if(FRGMLSTRK_CalculateValues.ano[k]>0)
                    ypoint = 100-(float)( ( 50 * FRGMLSTRK_CalculateValues.ano[k] / posnum ) );
                else
                    ypoint = 100+(float)( ( 200 * FRGMLSTRK_CalculateValues.ano[k] / maxY ) );
                if(FRGMLSTRK_CalculateValues.input_nob_gob[k]>0)
                    gypoint = 100-(float)( ( 50 * FRGMLSTRK_CalculateValues.input_nob_gob[k] / posnum ) );
                else
                    gypoint = 100+(float)( ( 200 * FRGMLSTRK_CalculateValues.input_nob_gob[k] / maxY ) );

                g2.setColor(Color.BLACK);
                g2.draw(new Line2D.Float(prevx, prevy, 215-strike1 + xpoint,  ypoint  ));

                g2.setColor(Color.BLUE);
                g2.setFont(new Font("Arial", 20, 40));
                g2.drawString(".",  215-strike1+xpoint - 6 ,  gypoint+3 );

                g2.setFont(new Font("Arial", 20, 12));
                g2.setColor(Color.black);
                prevx = 215-strike1 + xpoint;
                prevy = ypoint ;

            }

        }


        public void drawDepth(Graphics2D g2) {

            DecimalFormat df = new DecimalFormat("0.#");
            float zpoint1 = 0;
            float zpoint = 0;
            g2.setColor(Color.red);
            g2.fill(new Rectangle2D.Float(151, 300, 450, 250 ));
            g2.fillRect(150, 300,450, 250);
            for (float j = 300; j <= 550; j++){
                g2.draw(new Line2D.Float(600, j, 600+65, j-40 ));
            }
            Color col[] = {Color.BLACK,Color.BLACK,Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,
                Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,
```

```
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
olor.PINK,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
olor.PINK,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
olor.PINK,Color.WHITE};

        Color col1[] =
{Color.BLACK,Color.BLACK,Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAG
ENTA,Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA};

        for (int i = 2; i < FRGMLSTRK_CalculateValues.d_dep_di_arr.length; i++) {

            if (FRGMLSTRK_CalculateValues.d_den_for_arr[i-1] <= 2.38)
                g2.setColor(col[i]);
            else
                g2.setColor(col1[i]);

            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            zpoint1 = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i-1] / maxZ);

            if (i == 2) {
                g2.fill(new Rectangle2D.Float(151, 301, 450, zpoint));
                for (float j = 301;j<= 300 + zpoint; j++){
                    g2.draw(new Line2D.Float(600, j, 600+65, j-40));
                }
                g2.setColor(Color.BLACK);
                g2.setFont(new Font("Arial", 40,10));
                g2.drawString("----->"+df.format(FRGMLSTRK_CalculateValues.d_dep_di_arr[i]
"(km)", 600 + 65, 300 + zpoint - 38);
            }
            else if (i > 2) {

                g2.fill(new Rectangle2D.Float(151, 300 + zpoint1, 450, zpoint - zpoint1 ))
                for (float j = 300 + zpoint1;j <= 300 + zpoint; j++){
                    g2.draw(new Line2D.Float(600, j, 600 + 65, j - 40));
                }

                g2.setColor(Color.BLACK);
                g2.setFont(new Font("Arial", 40, 10));
                g2.drawString("----->" +
df.format(FRGMLSTRK_CalculateValues.d_dep_di_arr[i])+"(km)", 600 +65, 300 +
zpoint - 38);

            }

        }

        g2.setColor(Color.LIGHT_GRAY);
        for (float j = 151;j <= 600;j++){

            g2.draw(new Line2D.Float(j, 300, j + 65, 300 - 40));
        }
        g2.setColor(Color.BLACK);
        g2.draw(new Line2D.Float(215 - strike1, 260 + strike2, (float) 600 + 65 - strike1,
260 + strike2));
        g2.draw(new Line2D.Float(600, 300, 600, 300+zpoint));
        g2.draw(new Line2D.Float(600 + 65, 300 - 40, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(600, 300 + zpoint, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(150, 550, 600, 550));
        g2.setColor(Color.BLACK);

        for (int i = 188; i <= 600 + 32; i++){

            g2.drawString("-", i, 280 + 4);
            i = i + 4;
        }
```

```java
        g2.drawLine(90, 30, 950, 30);
        g2.drawLine(90, 565, 950, 565);
        g2.drawLine(950, 30, 950, 565);
        g2.drawLine(90, 30, 90, 565);
        g2.setFont(new Font("Arial", 40,20));
        g2.drawString("Interactive Gravity Modeling of Strike Limited Listric Fault Sources", 185, 20);
        g2.setFont(new Font("Arial", 40,12));
    }

    public void plotZCoordinates (Graphics2D g2) {

        i_no_obs = FRGMLSTRK_CalculateValues.i_no_obs;
        obs = new double[i_no_obs+1];
        for (int i = 1; i <= i_no_obs; i++) {

            obs[i] = FRGMLSTRK_CalculateValues.d_dis_km[i];

        }
        maxX = (float) obs[i_no_obs];
        maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
        float s = 0;
        float fc = 0;
        float spoint = 300;
        float fcpoint = (float)( 150 + ( 450 * FRGMLSTRK_CalculateValues.d_cftnt_arr[1] / maxX ) );
        float xpoint = 0;
        float zpoint = 0;

        while (s <= FRGMLSTRK_CalculateValues.d_max_dep) {

            float z1 = (float) 0.001;
            s = s + z1;
            fc = 0;
            for (int i = 1; i<FRGMLSTRK_CalculateValues.d_cftnt_arr.length; i++){

                fc = (float) (fc + FRGMLSTRK_CalculateValues.d_cftnt_arr[i] * Math.pow(s, i - 1));

            }
            xpoint = (float)( 450 * fc / maxX);
            zpoint = (float)( 250 * s / maxZ);

            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(fcpoint, spoint,(float)150 + xpoint, 300 + zpoint));

            g2.setColor(Color.RED);
            g2.draw(new Line2D.Float(150, 300 + zpoint, (float) 150 + xpoint, 300 + zpoint));

            fcpoint = (float) 150 + xpoint;
            spoint = 300 + zpoint;
            if (fcpoint > (float) 150 + ((700 * obs[i_no_obs] / maxX))){
                for (float j = spoint; j <= 550; j++){
                    g2.setColor(Color.RED);
                    g2.draw(new Line2D.Float(150, j, (float)( 150 + ((450 * obs[i_no_obs] / maxX))), j));
                    g2.draw(new Line2D.Float((float)( 151 + ((450 * obs[i_no_obs] / maxX))), j, (float)( 151 + ((450 * obs[i_no_obs] / maxX))) + 65, j - 40));
                }
                break;
            }

        }
    }


    public void drawDen(Graphics2D g2){

        FRGMLSTRK_DensityGraph de = new FRGMLSTRK_DensityGraph();
        de.denIndex(g2);
        g2.setFont(new Font("Arial", 20, 12));
        double denMax1 = FRGMLSTRK_CalculateValues.input_max_den - FRGMLSTRK_CalculateValues.input_min_den;
        float denXpoint = 0;
```

```java
        float denXpoint1 = 0;
        double denpoint1 = 0;
        float z1 = 0;
        float z2 = 0;
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_den_for_arr.length; i++){

            double denpoint = FRGMLSTRK_CalculateValues.d_den_for_arr[i] -
FRGMLSTRK_CalculateValues.input_min_den;
            if (i <= FRGMLSTRK_CalculateValues.d_den_for_arr.length - 2)
                denpoint1 = FRGMLSTRK_CalculateValues.d_den_for_arr[i + 1] -
FRGMLSTRK_CalculateValues.input_min_den;
            denXpoint = (float) (90 * denpoint / denMax1);
            denXpoint1 = (float) (90 * denpoint1 / denMax1);

            z1 = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i + 1] / maxZ);
            z2  = (float) (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            g2.setColor(Color.black);
            g2.draw(new Line2D.Float(820 + denXpoint, 300 + z2, 820 + denXpoint, 300 + z1))
            g2.draw(new Line2D.Float(820 + denXpoint, 300 + z1, 820 + denXpoint1, 300 +
z1));
        }
        g2.drawLine(820, 300, 910, 300);
        g2.drawLine(820, 300, 820, 550);
        g2.setColor(Color.white);
        g2.drawLine(821, 550, 910, 550);
    }


    public void idex(Graphics2D g){

        g.setColor(Color.BLUE);
        g.setFont(new Font("Arial", 20, 50));
        g.drawString(" ...     ", 750, 67);
        g.setFont(new Font("Arial", 20, 12));
        g.drawString(": Observed anomalies", 820, 70);
        g.setColor(Color.BLACK);
        g.drawString("_____", 765, 85);
        g.drawString(":  Modeled anomalies", 820, 90);
    }

}
---------------------------------------------------------------------------------------------

package com.frgmlstrk.view.event;

import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_PlainGraph;

public class FRGMLSTRK_PlotFault extends Applet{



    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static double val[] ;
    public static double val1[];
    public static int pos;
    public static MouseListener ml;
    com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new
com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
    FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();

    public void paint(Graphics g){

        FRGMLSTRK_MainPanel.p_Center.removeAll();
        FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);

```

```java
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.im);
            FRGMLSTRK_CalculateValues.len=0;
            val = new double[50] ;
            val1 = new double[50] ;
            Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.im.getGraphics();
            g2.setFont(new Font("Arial", 40,12));
            g2.setColor(Color.red);
            FRGMLSTRK_MainPanel.im.setEditable(false);
            FRGMLSTRK_MainPanel.im.setBackground(Color.WHITE);
            FRGMLSTRK_MainPanel.p_Center.validate();

            for (int j = 0; j < val.length; j++){
                val[j] = 0;
                val1[j] = 0;
            }
            pos = 2;
            cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues());
            Graphics2D g1 = (Graphics2D)FRGMLSTRK_MainPanel.im.getGraphics();
            pg.drawPlainGraph(g1);

            ml = new MouseAdapter(){
                public void mouseClicked(MouseEvent e) {
                    DecimalFormat f = new DecimalFormat("0.##");
                    float polyx = e.getX();
                    float polyy = e.getY();
                    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                    float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                    float x[] = new float[50];
                    float z[] = new float[50];
                    int get = pos;
                    FRGMLSTRK_CalculateValues.len = get;
                    Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.im.getGraphics();
                    if (polyy >= 300 && polyy <= (300 + zer) && polyx >= 150 && polyx <= 600){

                        FRGMLSTRK_PlotFault.val[get] = (float) Math.abs((FRGMLSTRK_PlainGraph.maxX1 * (150 - polyx)) / 450);
                        FRGMLSTRK_PlotFault.val1[get] = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300 - polyy) / zer);
                        x[get] = polyx;
                        z[get] = polyy;
                        g2.setColor(Color.white);
                        g2.draw(new Line2D.Float(150, polyy, 600, polyy ) );
                        g2.draw(new Line2D.Float(600, polyy, 600 + 65, polyy - 40 ) );
                        g2.fillRect(730, 160, 250, 40);

                        g2.setColor(Color.BLACK);
                        g2.drawString("x", polyx, polyy);
                        g2.drawString("("+f.format(FRGMLSTRK_PlotFault.val[get])+","+f.format(FRGMLSTRK_PlotFault.val1[get])+")", polyx, polyy);
                        g2.setFont(new Font("Arial", 40, 20));
                        if(get<=FRGMLSTRK_CalculateValues.i_d_poly)
                            g2.setColor(Color.red);
                        else
                            g2.setColor(Color.blue);

                        g2.drawString("Number of coordinates "+get, 740, 180);
                        g2.setFont(new Font("Arial", 40, 12));
                        FRGMLSTRK_PlotFault.pos++;
                    }

                    pg.drawPlainGraph(g2);
                    for (int i = 2; i <= FRGMLSTRK_CalculateValues.len; i++){
                        g2.setColor(Color.BLACK);
                        g2.drawString("("+f.format(FRGMLSTRK_PlotFault.val[FRGMLSTRK_CalculateValues.len])+","+f.format(FRGMLSTRK_PlotFault.val1[FRGMLSTRK_CalculateValues.len])+")", x[get], z[get]);
                    }
                    repaint();
                }

                public void mouseMoved(MouseEvent e){

                    DecimalFormat f = new DecimalFormat("0.##");
                    float xer = e.getX();
                    float yer = e.getY();
```

```
                    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                    float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                    float xpoint =(float) Math.abs((FRGMLSTRK_PlainGraph.maxX1 * (150 - xer)) /
450);
                    float zpoint = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300
yer) / zer);
                    if (yer >= 300 && yer <= (300 + zer) && xer >= 150 && xer <= 600){
                        Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.im.getGraphics();
                        g2.setColor(Color.black);
                        g2.drawString("X-coordinates            Z-coordinates", 780, 320);
                        g2.setColor(Color.red);
                        g2.fillRect(780, 320, 180, 20);
                        g2.setColor(Color.black);
                        g2.drawString(""+f.format(xpoint)+"
"+f.format(zpoint)+"", 780, 330);
                    }

                }

            };

            FRGMLSTRK_MainPanel.im.addMouseListener(ml);
            FRGMLSTRK_MainPanel.im.addMouseMotionListener((MouseMotionListener)ml);
            g2.setFont(new Font("Arial", 40, 20));
            g2.setColor(Color.black);
            g2.drawLine(615, 0, 615, 125);
            g2.drawLine(615, 125, 990, 125);
            g2.drawString("Instructions", 730, 20);
            g2.drawString("_____________", 710, 20);
            g2.setFont(new Font("Arial", 40, 12));
            g2.setColor(Color.red);
            g2.drawString("1) Select points on the fault plane by clicking the mouse ", 640,
40);
            g2.drawString("in the structure panel ", 644, 60);
            g2.setColor(Color.blue);
            g2.drawString("X", 640, 120);
            g2.setColor(Color.red);
            g2.drawString(":is origin of the fault plane  ", 650, 120);
            g2.setColor(Color.red);
            g2.drawString("2) Select the points within the specified boundary ", 640, 80);
            g2.drawString("of the structure panel", 644, 100);
        }

}
---------------------------------------------------------------------------------------------

package com.frgmlstrk.view.event;

import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_PlainGraph;

public class FRGMLSTRK_PlotDepth  {

    public static double val2[]= new double [50];

    public static int pos1;
    public static MouseListener ml2;
    com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new
com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
    com.frgmlstrk.view.FRGMLSTRK_DrawGraph dg = new
com.frgmlstrk.view.FRGMLSTRK_DrawGraph();
    FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();


    public void paint(Graphics g){

        FRGMLSTRK_MainPanel.p_Center.removeAll();
        FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);
        FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.img2);

```

```
            FRGMLSTRK_MainPanel.img2.setEditable(false);
            FRGMLSTRK_MainPanel.img2.setBackground(Color.WHITE);
            FRGMLSTRK_MainPanel.p_Center.validate();
            cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues());
            FRGMLSTRK_CalculateValues.inter = 2 + FRGMLSTRK_CalculateValues.i_nob_di;
            val2 = new double[50] ;

            final FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();
            try{
                cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues());
                cv.getCoefficients();
                Graphics2D g1 = (Graphics2D)FRGMLSTRK_MainPanel.img2.getGraphics();
                pg.drawPlainGraph(g1);
                dg.plotZCoordinates(g1);
                pg.drawPlainGraph(g1);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img2.getGraphics();
                for (int j = 0; j < val2.length; j++){
                    val2[j] = 0;

                }

                val2[1] = 0;
                val2[FRGMLSTRK_CalculateValues.inter] = FRGMLSTRK_CalculateValues.d_max_dep;
                pos1 = 2;

                ml2 = new MouseAdapter(){
                    public void mouseClicked(MouseEvent e) {

                        DecimalFormat f = new DecimalFormat("0.#");
                        float polyx = e.getX();
                        float polyy = e.getY();
                        float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                        float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                        float x[] = new float[50];
                        float z[] = new float[50];
                        int get = pos1;

                        Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img2.getGraphics();
                        if (polyy >= 300 && polyy <= (300 + zer) && polyx >= 150 && polyx <=
        600){
                            if (get < FRGMLSTRK_CalculateValues.inter){

                                val2[get] = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_de
        * (300 - polyy) / zer);
                                String alter = f.format(FRGMLSTRK_PlotDepth.val2[get]);
                                try {
                                    val2[get]= FRGMLSTRK_Utility.convertDouble(alter);
                                } catch (Exception e1) {
                                    e1.printStackTrace();
                                }
                                x[get] = polyx;
                                g2.setColor(Color.white);
                                g2.fillRect(730, 160, 250, 40);
                                g2.setColor(Color.BLACK);
                                g2.drawString("x", polyx, polyy);
                                g2.drawString("("+f.format(FRGMLSTRK_PlotDepth.val2[get])+")",
        polyx, polyy);
                                g2.setFont(new Font("Arial", 40, 20));
                                g2.drawString("Depths to be plotted
        "+Math.abs(get-(FRGMLSTRK_CalculateValues.inter-1)), 740, 180);
                                g2.setFont(new Font("Arial", 40, 12));
                            }

                            FRGMLSTRK_PlotDepth.pos1++;
                        }
                        pg.drawPlainGraph(g2);
                        dg.plotZCoordinates(g2);
                        pg.drawPlainGraph(g2);
                        for (int i = 2; i <= get; i++){
                            g2.setColor(Color.BLACK);
                            g2.drawString("("+f.format(FRGMLSTRK_PlotDepth.val2[get])+")",
        x[get], z[get]);
                        }
                        FRGMLSTRK_MainPanel.img2.repaint();
                    }
```

```java

                    public void mouseMoved(MouseEvent e){

                        DecimalFormat f = new DecimalFormat("0.#");
                        float xer = e.getX();
                        float yer = e.getY();
                        float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                        float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                        float zpoint = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep *
(300 - yer) / zer);
                        if (yer >= 300 && yer <= (300 + zer) && xer >= 150 && xer <=  600){
                            Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img2.getGraphics();
                            g2.setColor(Color.red);
                            g2.fillRect(780, 120, 100, 20);
                            g2.setColor(Color.black);
                            g2.drawString("Depth(km)", 800, 115);
                            g2.drawString(""+f.format(zpoint)+"", 810, 130);

                        }

                    }

                };

                FRGMLSTRK_MainPanel.img2.addMouseListener(ml2);
                FRGMLSTRK_MainPanel.img2.addMouseMotionListener((MouseMotionListener)ml2);
                g2.setFont(new Font("Arial", 40, 20));
                g2.setColor(Color.black);
                g2.drawString("Depths to be plotted "+(FRGMLSTRK_CalculateValues.inter-2), 740,
180);
                g2.drawLine(615, 0, 615, 85);
                g2.drawLine(615, 85, 990, 85);
                g2.drawString("Instructions", 730, 20);
                g2.drawString("_____________", 710, 20);
                g2.setFont(new Font("Arial", 40, 12));
                g2.setColor(Color.red);
                g2.drawString("1) Select the depth values by clicking the mouse  ", 660, 40);
                g2.drawString("in the structure panel.",664,60);
                g2.setColor(Color.red);
                g2.drawString("2) Click the mouse in ascending order of depth", 660, 80);
            }

            catch(Exception e){
                //exception
            }
        }
}
------------------------------------------------------------------------------------------

package com.frgmlstrk.view.event;

import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_DensityGraph;
import com.frgmlstrk.view.graph.FRGMLSTRK_DepthLines;
import com.frgmlstrk.view.graph.FRGMLSTRK_PlainGraph;


public class FRGMLSTRK_PlotDensity {

    public static double val3[];
    public static int pos2;
    public static MouseListener ml4;
    com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new
com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
    com.frgmlstrk.view.FRGMLSTRK_DrawGraph dg = new
com.frgmlstrk.view.FRGMLSTRK_DrawGraph();
    FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();
    FRGMLSTRK_DensityGraph de = new FRGMLSTRK_DensityGraph();
    FRGMLSTRK_DepthLines dl = new FRGMLSTRK_DepthLines();

```

```java

    public void paint(Graphics g){

        val3 = new double[50] ;
        FRGMLSTRK_MainPanel.p_Center.removeAll();
        FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);
        FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.den);
        FRGMLSTRK_MainPanel.den.setEditable(false);
        FRGMLSTRK_MainPanel.den.setBackground(Color.WHITE);
        FRGMLSTRK_MainPanel.p_Center.validate();
        try{
            com.frgmlstrk.view.FRGMLSTRK_MainPanel.clearPanel(FRGMLSTRK_MainPanel.den);
            cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues());
            FRGMLSTRK_CalculateValues.inter = 2 + FRGMLSTRK_CalculateValues.i_nob_di;
            cv.getCoefficients();
            cv.depthValues();
            Graphics2D g1 = (Graphics2D)FRGMLSTRK_MainPanel.den.getGraphics();
            pg.drawPlainGraph(g1);
            dl.drawPlaneLines(g1);
            dg.plotZCoordinates(g1);
            pg.drawPlainGraph(g1);
            de.drawPlain(g1);
            Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.den.getGraphics();

            for (int j = 0; j < val3.length; j++){
                val3[j] = 0;
            }
            pos2 = 1;
            ml4 = new MouseAdapter(){
                public void mouseClicked(MouseEvent e) {

                    float diff = (float) (FRGMLSTRK_CalculateValues.input_max_den - FRGMLSTRK_CalculateValues.input_min_den);
                    float denx = e.getX();
                    float deny = e.getY();
                    float maxZ = diff;
                    float zer = (float) (90 * diff / maxZ);
                    float []y = new float[50];
                    float []x = new float[50];
                    int get = pos2;

                    DecimalFormat f = new DecimalFormat("0.##");
                    Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.den.getGraphics();
                    if (deny >= 300 && deny <= 550 && denx >= 820 && denx <= 910){
                        if (get < FRGMLSTRK_CalculateValues.inter){

                            val3[get] = (float)(FRGMLSTRK_CalculateValues.input_min_den + Math.abs(diff * (820 - denx) / zer));
                            String alter = f.format(FRGMLSTRK_PlotDensity.val3[get]);
                            try {
                                val3[get] = FRGMLSTRK_Utility.convertDouble(alter);
                            } catch (Exception e1) {
                                e1.printStackTrace();
                            }
                            y[get] = deny;
                            x[get] = denx;
                            g2.setColor(Color.white);
                            g2.fillRect(730, 160, 250, 40);
                            g2.setColor(Color.BLACK);
                            g2.drawString("x", denx, deny);
                            g2.drawString("("+f.format(FRGMLSTRK_PlotDensity.val3[get])+")' denx, deny);
                            g2.setFont(new Font("Arial", 40, 20));
                            g2.drawString("Densities to be plotted "+Math.abs(get-(FRGMLSTRK_CalculateValues.inter-1)), 740, 180);
                            g2.setFont(new Font("Arial", 40, 12));
                        }

                        FRGMLSTRK_PlotDensity.pos2++;
                    }


                    dl.drawPlaneLines(g2);
                    dg.plotZCoordinates(g2);
                    pg.drawPlainGraph(g2);
```

```
                    de.drawPlain(g2);
                    for (int i = 1;i <= get; i++){

                        g2.setColor(Color.BLACK);
                        g2.drawString("("+f.format(FRGMLSTRK_PlotDensity.val3[get])+")", x[get], y[get]);
                    }
                    FRGMLSTRK_MainPanel.den.repaint();
                }

                public void mouseMoved(MouseEvent e){

                    float diff = (float) (FRGMLSTRK_CalculateValues.input_max_den - FRGMLSTRK_CalculateValues.input_min_den);
                    DecimalFormat f = new DecimalFormat("0.##");
                    float xer = e.getX();
                    float yer = e.getY();
                    float maxZ = diff;
                    float zer = (float) (90 * diff / maxZ);
                    float zpoint = (float) Math.abs(diff * (820 - xer) / zer);
                    if (yer >= 300 && yer <= 550 && xer >= 820 && xer <= 910){
                        Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.den.getGraphics();
                        g2.setColor(Color.red);
                        g2.fillRect(780, 120, 100, 20);
                        g2.setColor(Color.black);
                        g2.drawString("Density range", 800, 115);
                        g2.drawString(""+f.format(FRGMLSTRK_CalculateValues.input_min_den+zpoint)+"", 810, 130);
                    }
                }

            };
            FRGMLSTRK_MainPanel.den.addMouseListener(ml4);
            FRGMLSTRK_MainPanel.den.addMouseMotionListener((MouseMotionListener)ml4);

            g2.setFont(new Font("Arial", 40, 20));
            g2.setColor(Color.black);
            g2.drawString("Densities to be plotted "+(FRGMLSTRK_CalculateValues.inter-1), 740, 180);
            g2.drawLine(615, 0, 615, 85);
            g2.drawLine(615, 85, 990, 85);
            g2.drawString("Instructions", 730, 20);
            g2.drawString("_____________", 710, 20);
            g2.setFont(new Font("Arial", 40, 12));
            g2.setColor(Color.red);
            g2.drawString("1) Select the density values by clicking the mouse  ", 642, 40);
            g2.drawString("in the density-depth panel ", 646, 60);
            g2.setColor(Color.red);
            g2.drawString("2) Click the mouse between two consecutive depth interfaces", 642, 80);
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
------------------------------------------------------------------------------------------

package com.frgmlstrk.view.event;

import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_HandleException;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_PlainGraph;

public class FRGMLSTRK_EditFault {
    public static float zpoint = 0;
    public static int count = 0;
    public static MouseListener ml1;
    com.frgmlstrk.view.FRGMLSTRK_DrawGraph dg = new com.frgmlstrk.view.FRGMLSTRK_DrawGraph();
```

```
        FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();
        com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
        public void paint(Graphics g){
            final DecimalFormat f = new DecimalFormat("0.##");
            FRGMLSTRK_MainPanel.p_Center.removeAll();
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.fl);
            FRGMLSTRK_MainPanel.fl.setEditable(false);
            FRGMLSTRK_MainPanel.fl.setBackground(Color.WHITE);
            FRGMLSTRK_MainPanel.p_Center.validate();
            cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues());
            try {
                cv.getCoefficients();
            } catch (FRGMLSTRK_HandleException e2) {

                e2.printStackTrace();
            }
            Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.fl.getGraphics();

            pg.drawPlainGraph(g2);
            dg.plotZCoordinates(g2);
            pg.drawPlainGraph(g2);
            g2.setColor(Color.black);
            for(int i = 1;i<=FRGMLSTRK_CalculateValues.len;i++){
                float xpoint  = (float)(450*FRGMLSTRK_CalculateValues.d_x_km_arr[i]/FRGMLSTRK_PlainGraph.maxX1);
                float zpoint  = (float)(250*FRGMLSTRK_CalculateValues.d_z_km_arr[i]/FRGMLSTRK_CalculateValues.d_max_dep);
                g2.drawString("("+f.format(FRGMLSTRK_CalculateValues.d_x_km_arr[i])+","+f.forma(FRGMLSTRK_CalculateValues.d_z_km_arr[i])+")", 160+xpoint, 304+zpoint);
                g2.drawString("X", 150+xpoint-3, 304+zpoint);
            }

            ml1 = new MouseAdapter(){

                public void mouseDragged(MouseEvent e1) {
                    DecimalFormat f = new DecimalFormat("0.##");
                    float x2 = e1.getX();
                    float y2 = e1.getY();
                    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                    float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                    Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.fl.getGraphics();
                    if (y2 > 300 && y2 <= (300 + zer) && x2 <= 600){

                        g2.setColor(Color.red);
                        g2.fillRect(750, 180, 180, 30);
                        g2.setColor(Color.BLACK);
                        zpoint = (float) Math.abs((FRGMLSTRK_PlainGraph.maxX1 * (150 - x2)) / 450);
                        g2.drawString("Sl.no                        x-coordinate(km)", 750, 180);
                        g2.drawString(""+f.format(zpoint), 870, 200);
                        g2.setColor(Color.black);
                        g2.drawString(""+(count ), 770, 200);

                    }

                    pg.drawPlainGraph(g2);
                    dg.plotZCoordinates(g2);
                    pg.drawPlainGraph(g2);

                }


                public void mousePressed(MouseEvent e1) {

                    float x1 = e1.getX();
                    float y1 = e1.getY();
                    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                    float diff = (float)(FRGMLSTRK_CalculateValues.d_dis_km[FRGMLSTRK_CalculateValues.i_no_obs]/100);
                    float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                    float xer = (float) Math.abs((FRGMLSTRK_PlainGraph.maxX1 * (150 - (x1-3)))
```

```
450);

                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.fl.getGraphics();
                if (y1 > 300 && y1 < (300 + zer) && x1 <= 600){


                    for (int kk = 2; kk <= FRGMLSTRK_CalculateValues.len; kk++){

                        if (Math.abs((float)FRGMLSTRK_CalculateValues.d_x_km_arr[kk] - xer
)<= diff || Math.abs((float)FRGMLSTRK_CalculateValues.d_z_km_arr[kk]
- zer )<= 0 ){
                            count = kk;

                        }
                        g2.setColor(Color.white);
                        g2.drawString(""+count, 750, 380);

                    }

                }

            }

            public void mouseReleased(MouseEvent e1) {

                float x = e1.getX();
                float y = e1.getY();
                float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.fl.getGraphics();
                FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();
                if (y > 300 && y <= (300 + zer) && x <= 600){

                    float xer = (float) Math.abs((FRGMLSTRK_PlainGraph.maxX1 * (150 - x)) /
450);
                    FRGMLSTRK_PlotFault.val[count]=xer;
                    FRGMLSTRK_CalculateValues.d_x_km_arr[count] = xer;

                }

                try {

                    cv.getCoefficients();
                } catch (FRGMLSTRK_HandleException e) {
                    e.printStackTrace();
                }
                FRGMLSTRK_MainPanel.clearPanel(FRGMLSTRK_MainPanel.fl);

                Graphics2D g1 = (Graphics2D)FRGMLSTRK_MainPanel.fl.getGraphics();
                pg.drawPlainGraph(g1);
                dg.plotZCoordinates(g1);
                g1.setColor(Color.black);
                for(int i = 1;i<=FRGMLSTRK_CalculateValues.len;i++){
                    float xpoint  = (float)
(450*FRGMLSTRK_CalculateValues.d_x_km_arr[i]/FRGMLSTRK_PlainGraph.maxX1)
;
                    float zpoint  = (float)(250*
FRGMLSTRK_CalculateValues.d_z_km_arr[i]/FRGMLSTRK_CalculateValues.d_max_
dep);
                    g2.drawString("("+f.format(FRGMLSTRK_CalculateValues.d_x_km_arr[i])+","
f.format(FRGMLSTRK_CalculateValues.d_z_km_arr[i])+")", 160+xpoint,
304+zpoint);
                    g2.drawString("X", 150+xpoint-3, 304+zpoint);
                }
            }

        };

        FRGMLSTRK_MainPanel.fl.addMouseListener(ml1);
        FRGMLSTRK_MainPanel.fl.addMouseMotionListener((MouseMotionListener)ml1);
    }
}
--------------------------------------------------------------------------------------------
```

```java
package com.frgmlstrk.view.event;

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_DensityGraph;
import com.frgmlstrk.view.graph.FRGMLSTRK_DepthLines;


public class FRGMLSTRK_EditDepth {

    public static float zpoint = 0;
    public static int count = 0;
    public static MouseListener ml1;
    com.frgmlstrk.view.FRGMLSTRK_DrawGraph dg = new com.frgmlstrk.view.FRGMLSTRK_DrawGraph();
    FRGMLSTRK_DepthLines dl = new FRGMLSTRK_DepthLines();

    public void paint(Graphics g){
        ml1 = new MouseAdapter(){
            DecimalFormat f = new DecimalFormat("0.#");
            public void mouseDragged(MouseEvent e1) {

                float x2 = e1.getX();
                float y2 = e1.getY();
                float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                if (y2 > 300 && y2 <= (300 + zer) && x2 <= 600){

                    g2.setColor(Color.red);
                    g2.fillRect(750, 180, 180, 30);
                    g2.setColor(Color.BLACK);
                    zpoint = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300 - y2) / zer);
                    g2.drawString("Interface                        Depth", 750, 180);
                    g2.drawString(""+f.format(zpoint), 870, 200);
                    g2.setColor(Color.black);
                    g2.drawString(""+(count - 1), 770, 200);

                }
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dl.drawDepthLine(g2);

            }


            public void mousePressed(MouseEvent e1) {

                float x1 = e1.getX();
                float y1 = e1.getY();
                float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                float dep = (float)FRGMLSTRK_CalculateValues.d_max_dep / 100;
                float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                if (y1 > 300 && y1 < (300 + zer) && x1 <= 600){

                    float zpoint1 = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300 - y1) / zer);
                    for (int kk = 2; kk < FRGMLSTRK_CalculateValues.inter; kk++){

                        if (Math.abs((float)FRGMLSTRK_CalculateValues.d_dep_di_arr[kk] - zpoint1) <= dep){
                            count = kk;
                        }
                        g2.setColor(Color.white);
                        g2.drawString(""+count, 750, 380);

                    }

```

```
                }
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dl.drawDepthLine(g2);

            }


            public void mouseReleased(MouseEvent e1) {

                float x = e1.getX();
                float y = e1.getY();
                float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                float zer = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();

                if (y > 300 && y <= (300 + zer) && x <= 600&&FRGMLSTRK_CalculateValues.inte > 2){

                    g2.setColor(Color.BLACK);
                    g2.draw(new Line2D.Float(150, y, 600, y ) );
                    g2.draw(new Line2D.Float(600, y, 600 + 65, y - 40 ) );
                    zpoint = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300 - y) / zer);
                    String alter = f.format(zpoint);
                    try{
                        FRGMLSTRK_PlotDepth.val2[count] = FRGMLSTRK_Utility.convertDouble(alter);;
                        FRGMLSTRK_CalculateValues.d_dep_di_arr[count] = FRGMLSTRK_Utility.convertDouble(alter);;
                    }
                    catch(Exception e){

                    }
                    g2.drawString(""+f.format(zpoint), 600, y);
                }
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dl.drawDepthLine(g2);

                com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
                cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues())
                try {
                    cv.cal();
                    com.frgmlstrk.view.FRGMLSTRK_TableView.populateEastPanel(FRGMLSTRK_CalclateValues.obj);
                    drawGraph(FRGMLSTRK_MainPanel.img3);

                } catch (Exception e) {

                }
            }

        };

        FRGMLSTRK_MainPanel.img3.addMouseListener(ml1);
        FRGMLSTRK_MainPanel.img3.addMouseMotionListener((MouseMotionListener)ml1);
    }
    public void drawGraph(TextArea t){

        FRGMLSTRK_DensityGraph de = new FRGMLSTRK_DensityGraph();
        Graphics2D g2 = (Graphics2D)t.getGraphics();//(Graphics2D)g1 ;
        com.frgmlstrk.view.FRGMLSTRK_MainPanel.clearPanel(t);
        dg.plot(g2);
        dg.plotXYCoordinates(g2);
        dg.drawGraph(g2);
        dl.drawDepthLine(g2);
        dg.plotZCoordinates(g2);
        dg.plot(g2);
        de.Plainden(g2);
    }
}
--------------------------------------------------------------------------------------------
```

```
package com.frgmlstrk.view.event;

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_DensityGraph;

public class FRGMLSTRK_EditDensity {

    MouseListener ml5;
    public static float dpoint = 0;
    public static int count1 = 0;
    FRGMLSTRK_DensityGraph de = new FRGMLSTRK_DensityGraph();

    public void paint(Graphics g){

        ml5 = new MouseAdapter(){

            public void mouseDragged(MouseEvent e1) {

                float diff = (float) (FRGMLSTRK_CalculateValues.input_max_den -
FRGMLSTRK_CalculateValues.input_min_den);
                float x2 = e1.getX();
                float y2 = e1.getY();
                float maxZ = diff;
                float zer = (float) (90 * diff / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                if (y2 > 300 && y2 <= 570 && x2 >= 820 && x2 <= 910){
                    g2.setColor(Color.red);
                    g2.fillRect(750, 240, 180, 30);
                    g2.setColor(Color.black);
                    dpoint = (float)(FRGMLSTRK_CalculateValues.input_min_den + Math.abs(dif
* (820 - x2) / zer));
                    DecimalFormat d = new DecimalFormat("0.##");
                    g2.drawString("Formation                        Density", 750, 235);
                    g2.drawString(""+d.format(dpoint), 870, 250);
                    g2.drawString(""+count1, 770, 250);

                }
                de.Plainden(g2);

            }


            public void mousePressed(MouseEvent e1) {

                float x1 = e1.getX();
                float y1 = e1.getY();

                float maxZ1 = (float)FRGMLSTRK_CalculateValues.d_max_dep;
                float zer1 = (float) (250 * FRGMLSTRK_CalculateValues.d_max_dep / maxZ1);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                if (y1 > 300 && y1 < 570 && x1 >= 820 && x1 <= 910){

                    float zp = (float) Math.abs(FRGMLSTRK_CalculateValues.d_max_dep * (300
y1) / zer1);
                    for (int kk = 1; kk < FRGMLSTRK_CalculateValues.inter; kk++){
                        if (zp > FRGMLSTRK_CalculateValues.d_dep_di_arr[kk] && zp <
FRGMLSTRK_CalculateValues.d_dep_di_arr[kk + 1]){
                            count1 = kk;
                        }
                        g2.setColor(Color.white);
                        g2.drawString(""+count1, 750, 380);

                    }
                }
                de.Plainden(g2);
            }

```

```
            public void mouseReleased(MouseEvent e1) {

                float diff = (float) (FRGMLSTRK_CalculateValues.input_max_den - FRGMLSTRK_CalculateValues.input_min_den);
                DecimalFormat d = new DecimalFormat("0.##");
                float x = e1.getX();
                float y = e1.getY();
                float maxZ = diff;
                float zer = (float) (90 * diff / maxZ);
                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                if (y > 300 && y <= 570 && x >= 820 && x <= 910){

                    float zp = (float)(FRGMLSTRK_CalculateValues.input_min_den + (90 * (FRGMLSTRK_CalculateValues.d_den_for_arr[count1]-FRGMLSTRK_CalculateValues.input_min_den) / maxZ));
                    g2.setColor(Color.WHITE);
                    g2.draw(new Line2D.Float(820 + zp, y, 820 + zp, y));
                    g2.drawString(""+d.format(FRGMLSTRK_CalculateValues.d_den_for_arr[count]), 910, 300 + zp);
                    g2.setColor(Color.black);
                    dpoint = (float)(FRGMLSTRK_CalculateValues.input_min_den+ Math.abs(diff * (820 - x) / zer));
                    String alter = d.format(dpoint);
                    try {
                        FRGMLSTRK_PlotDensity.val3[count1] = FRGMLSTRK_Utility.convertDouble(alter);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    g2.drawString(""+d.format(dpoint), 910, y);
                }
                de.Plainden(g2);
            }

        };

        FRGMLSTRK_MainPanel.img3.addMouseListener(ml5);
        FRGMLSTRK_MainPanel.img3.addMouseMotionListener((MouseMotionListener)ml5);
    }
}
---------------------------------------------------------------------------------------------

package com.frgmlstrk.view.graph;

import java.awt.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_Utility;

public class FRGMLSTRK_PlainGraph {

    int i_no_obs;
    public static float maxX1;
    public void drawPlainGraph(Graphics2D g2){

        g2.setColor(Color.BLACK);
        g2.setFont(new Font("Arial", 20, 12));
        i_no_obs = FRGMLSTRK_CalculateValues.i_no_obs;
        double obser[] = new double[i_no_obs + 1];
        for (int i = 1; i <= i_no_obs; i++) {

            obser[i] = FRGMLSTRK_CalculateValues.d_dis_km[i];
        }
        maxX1 = (float) FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs];
        float maxZ1 = (float)FRGMLSTRK_CalculateValues.d_max_dep;
        float points = maxX1 / 5;
        int zInterval = 50;
        DecimalFormat f = new DecimalFormat("0.#");
        g2.drawLine(150,50,150,300);
        g2.drawString("|",(float) 598, 300);
        g2.drawString(""+f.format(FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs]),600, 320);
        int xInterval=90;
        for (int x = xInterval, j =1; x < 600; x+=xInterval)

```

```
            {
                if(j > 4)
                    break;
                g2.drawString("|",150+x,300);
                g2.drawString("" + (points*j), 147+x, 320);
                j++;
            }
            points = maxZ1 / 5 ;
            g2.drawString("0", 125, 305);
            DecimalFormat d = new DecimalFormat("0.#");
            for (int x = zInterval+250,j = 1; x < 550; x += zInterval){

                g2.drawString("-", 148, 52 + x);
                g2.drawString("" +d.format(points * j), 125, 50 + x);
                j++;
            }
            g2.draw(new Line2D.Float(150, 300, 600, 300));
            g2.draw(new Line2D.Float(150, 300, 150, 550));
            float maxY = (float) FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.input_nob_gob);
            double minAno = FRGMLSTRK_Utility.findMinimumNumber1(FRGMLSTRK_CalculateValues.input_nob_gob);
            double maxAno = FRGMLSTRK_Utility.findMaximumNumber(FRGMLSTRK_CalculateValues.input_nob_gob, minAno);

            if (minAno < 0 && maxAno > 0){

                float xpoint = 0;
                float gypoint = 0;
                g2.setFont(new Font("Arial", 20, 12));
                g2.setColor(Color.black);
                double store1[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
                double negstore1[] = new double[FRGMLSTRK_CalculateValues.i_no_obs+1];
                for(int i = 1; i <= FRGMLSTRK_CalculateValues.i_no_obs; i++){

                    if(FRGMLSTRK_CalculateValues.input_nob_gob[i]>0)
                        store1[i] = FRGMLSTRK_CalculateValues.input_nob_gob[i];
                    else
                        negstore1[i] = FRGMLSTRK_CalculateValues.input_nob_gob[i];
                }
                float posnum =  (float) FRGMLSTRK_Utility.findMaximumNumber1(store1);
                maxY = (float) FRGMLSTRK_Utility.findMaximumNumber(negstore1);
                DecimalFormat df = new DecimalFormat("0.##");
                g2.drawString("0",115,100);
                g2.drawString("-",148, 55);
                g2.drawString(""+df.format(posnum), 115, 50);

                points = maxY / 4;
                int yInterval=50;
                for (int x = yInterval, j = 1; x < 250; x+=yInterval){

                    g2.drawString("-", 148, 100 + x );
                    g2.drawString("" + f.format(points * j), 115, 100 + x );
                    j++;
                }
                for (int k = 1; k <= i_no_obs; k++) {

                    xpoint = (float)( 450 * obser[k] / maxX1);
                    if(FRGMLSTRK_CalculateValues.input_nob_gob[k]>0)
                        gypoint = 100-(float)( ( 50 * FRGMLSTRK_CalculateValues.input_nob_gob[k / posnum ) );
                    else
                        gypoint = 100+(float)( ( 200 * FRGMLSTRK_CalculateValues.input_nob_gob[k] / maxY ) );
                    g2.setColor(Color.BLUE);
                    g2.setFont(new Font("Arial", 20, 40));
                    g2.drawString(".",  150 + xpoint - 6 ,  gypoint + 3 );
                    g2.setFont(new Font("Arial", 20, 12));
                    g2.setColor(Color.black);

                }

            }
            else{
```

```java
            g2.drawString("0", 135, 50);
            points = (int)maxY / 5;

            int yInterval = 50;
            for (int x = yInterval, j = 1; x < 250; x += yInterval){

                g2.drawString("-", 148, 52 + x);
                g2.drawString("" + (points * j), 115, 50 + x);
                j++;
            }

            float xpoint = 0;
            float gypoint = 0;

            for (int k = 1; k <= i_no_obs; k++) {

                xpoint = (float)( 450 * obser[k] / maxX1);
                gypoint = (float)( ( 250 * FRGMLSTRK_CalculateValues.input_nob_gob[k] / max
) );

                g2.setColor(Color.BLUE);
                g2.setFont(new Font("Arial", 20, 40));
                g2.drawString(".",  150 + xpoint - 6 , 50 + gypoint + 3);

            }
        }
        double inipoint = calInip();
        float plotini = (float)( 450 * inipoint / maxX1);
        g2.setFont(new Font("Arial", 20, 12));
        g2.drawString("X", 150 + plotini-3 , 304);
        g2.draw(new Line2D.Float(600, 300, 600, 300 + (float)(250 *
FRGMLSTRK_CalculateValues.d_max_dep / maxZ1)));
        g2.draw(new Line2D.Float(150, 300 + (float)(250 *
FRGMLSTRK_CalculateValues.d_max_dep / maxZ1), 600, 300 + (float)(250 *
FRGMLSTRK_CalculateValues.d_max_dep / maxZ1)));
    }

    public double calInip(){
        double gmax = Math.abs(FRGMLSTRK_CalculateValues.input_nob_gob[1]);

        for (int k = 1; k <= i_no_obs; k++) {

            if (Math.abs(FRGMLSTRK_CalculateValues.input_nob_gob[k]) - gmax > 0)
                gmax = Math.abs(FRGMLSTRK_CalculateValues.input_nob_gob[k]);

        }
        double datum = FRGMLSTRK_CalculateValues.input_nob_gob[1];
        double r =
FRGMLSTRK_CalculateValues.input_nob_gob[i_no_obs]-FRGMLSTRK_CalculateValues.input_no
b_gob[1];
        int kk1 = 1;
        double gh = 0.5 * r;
        kk1 = kk1 + 1;
        double XH = 0;

        while ((( FRGMLSTRK_CalculateValues.input_nob_gob[kk1] - datum) / gh ) - 1.0 < 0) {

            kk1 = kk1 + 1;
        }

        if ((( FRGMLSTRK_CalculateValues.input_nob_gob[kk1] - datum) / gh) - 1.0 > 0) {

            XH = FRGMLSTRK_CalculateValues.d_dis_km[kk1-1] + ( ( gh + datum -
FRGMLSTRK_CalculateValues.input_nob_gob[kk1-1]) * (FRGMLSTRK_CalculateValues.
d_dis_km[kk1] - FRGMLSTRK_CalculateValues.d_dis_km[kk1-1] ) ) / (
FRGMLSTRK_CalculateValues.input_nob_gob[kk1] -
FRGMLSTRK_CalculateValues.input_nob_gob[kk1-1] );
        }
        if ((( FRGMLSTRK_CalculateValues.input_nob_gob[kk1] - datum ) / gh) - 1.0 == 0) {

            XH = FRGMLSTRK_CalculateValues.d_dis_km[kk1];
        }

        double ini = XH;
        return ini;
```

```java
    }
}
--------------------------------------------------------------------------------------------

package com.frgmlstrk.view.graph;

import java.awt.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.view.FRGMLSTRK_DrawGraph;

public class FRGMLSTRK_DepthLines {

    int i_no_obs = FRGMLSTRK_CalculateValues.i_no_obs;
    float maxX;
    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
    public void drawDepthLine(Graphics2D g2) {

        maxX = (float)FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs];
        float zpoint = 0;
        DecimalFormat d = new DecimalFormat("0.#");
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_dep_di_arr.length; i++) {

            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            g2.draw(new Line2D.Float(151, 300 + zpoint, 600, 300 + zpoint ));
            g2.draw(new Line2D.Float(600, 300 + zpoint, 600 + 65, 300 + zpoint - 40 ));
            if (i > 1 && i < FRGMLSTRK_CalculateValues.inter){
                g2.drawString(""+d.format(FRGMLSTRK_CalculateValues.d_dep_di_arr[i]), 600,
300 + zpoint);
                g2.drawString(""+(i - 1), 600 + 85, 300 + zpoint - 40);
            }
        }

        g2.setColor(Color.BLACK);
        g2.draw(new Line2D.Float(215 - FRGMLSTRK_DrawGraph.strike1, 260 +
FRGMLSTRK_DrawGraph.strike2, (float) 600 + 65 - FRGMLSTRK_DrawGraph.strike1, 260 +
FRGMLSTRK_DrawGraph.strike2));
        g2.draw(new Line2D.Float(600, 300, 600, 300 + zpoint));
        g2.draw(new Line2D.Float(600 + 65, 300 - 40, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(600, 300 + zpoint, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(150, 300, 150 + 65, 300 - 40));
        g2.draw(new Line2D.Float(150 + 65, 300 - 40, 600 + 65, 300 - 40));
        g2.draw(new Line2D.Float(600, 300, 600 + 65, 300 - 40));
        g2.setColor(Color.BLACK);

        for (int i= 188; i <= 600 + 32; i++){

            g2.drawString("-", i, 280 + 4);
            i = i + 4;

        }
    }

    public void drawPlaneLines(Graphics2D g){
        maxX =(float)FRGMLSTRK_CalculateValues.d_dis_km[i_no_obs];
        float zpoint = 0;
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_dep_di_arr.length; i++) {
            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            g.setColor(Color.black);
            g.draw(new Line2D.Float(150, 300 + zpoint, 600, 300 + zpoint ));

        }
    }
}
--------------------------------------------------------------------------------------------

package com.frgmlstrk.view.graph;

import java.awt.*;
import java.awt.geom.Line2D;
import java.text.DecimalFormat;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
```

```
public class FRGMLSTRK_DensityGraph {

    float maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
    public void drawPlain(Graphics2D g2){

        g2.setColor(Color.black);
        g2.drawLine(820,300,910,300);
        g2.drawLine(820, 300, 820,550);
        float zpoint = 0;
        denIndex(g2);
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_dep_di_arr.length; i++) {
            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            g2.setColor(Color.black);
            g2.draw(new Line2D.Float(820, 300 + zpoint, 910, 300 + zpoint ));

        }

    }

    public void Plainden(Graphics2D g2){

        denIndex(g2);

        g2.drawLine(820, 300, 910, 300);
        g2.drawLine(820, 300, 820, 550);
        float zpoint = 0;
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_dep_di_arr.length; i++) {
            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            g2.setColor(Color.black);
            g2.draw(new Line2D.Float(820, 300 + zpoint, 910, 300 + zpoint));

        }

        float diff = (float) (FRGMLSTRK_CalculateValues.input_max_den -
FRGMLSTRK_CalculateValues.input_min_den);
        float maxDen = diff;
        for (int i = 1; i < FRGMLSTRK_CalculateValues.d_den_for_arr.length; i++){
            zpoint = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i] / maxZ);
            float zpoint1 = (float) (250 * FRGMLSTRK_CalculateValues.d_dep_di_arr[i + 1] /
maxZ);
            float denpoint = (float) (90 * (FRGMLSTRK_CalculateValues.d_den_for_arr[i] -
FRGMLSTRK_CalculateValues.input_min_den) / maxDen);
            g2.setColor(Color.blue);
            g2.draw(new Line2D.Float(820 + denpoint, 300 + zpoint, 820 + denpoint, 300 +
zpoint1));
        }
    }

    public void denIndex(Graphics2D g){

        DecimalFormat d = new DecimalFormat("0.#");
        maxZ = (float)FRGMLSTRK_CalculateValues.d_max_dep;
        float points = maxZ / 5 ;
        int zInterval = 50;

        for (int x = zInterval + 250, j = 1; x < 550; x += zInterval){

            g.drawString("-", 818, 52 + x);
            g.drawString("" +d.format(points * j), 795, 50 + x);
            j++;
        }

        g.setColor(Color.red);
        String []b = {"D", "E", "P", "T", "H", "(k", "m)"};

        for (int i = 0; i < b.length; i++) {

            g.drawString(""+b[i], 765,  350 + ( i * 20 ) );
        }
        g.setColor(Color.black);
        g.drawString(""+FRGMLSTRK_CalculateValues.input_min_den, 818, 298);
        g.drawString(""+FRGMLSTRK_CalculateValues.input_max_den, 908, 298);
        g.setColor(Color.red);
        g.drawString("Density", 850, 280);
```

```
        g.drawString("(gm/cm  )",850 , 295);
        g.setFont(new Font("Arial", 20, 9));
        g.drawString("3",890 , 292);
        g.setFont(new Font("Arial", 20, 12));
        g.setColor(Color.black);
        g.drawString("0", 795, 305);

    }
}
------------------------------------------------------------------------------------------

package com.frgmlstrk.model;

import java.text.DecimalFormat;
import java.util.Arrays;
import java.util.HashMap;

import javax.swing.JFrame;
import javax.swing.JOptionPane;



import com.frgmlstrk.util.FRGMLSTRK_HandleException;
import com.frgmlstrk.util.FRGMLSTRK_Utility;
import com.frgmlstrk.view.FRGMLSTRK_TableView;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotDensity;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotDepth;
import com.frgmlstrk.view.event.FRGMLSTRK_PlotFault;


public class FRGMLSTRK_CalculateValues {

    public static Object obj[][] = null;
    public static int inter = 2;
    public static int i_no_obs, i_d_poly, i_nob_di,len = 0;
    public static double d_str_st, d_off_pro, d_max_dep, d_base_den, input_max_den,
input_min_den = 0;
    public static double d_dep_di_arr[], d_ele_km[], d_den_for_arr[], d_dis_km[], ano[],
d_cftnt_arr[], x[], cftnt[] = null;
    public static double input_nob_gob[] = null;
    public static String input_area_name, input_profile_num = "";
    public static double d_x_km_arr[], d_z_km_arr[];

    public void getAnamolyValues(HashMap h_Map) {

        try {
            i_no_obs = FRGMLSTRK_Utility.convertInteger((String)h_Map.get("N_OBS"));
            i_d_poly = FRGMLSTRK_Utility.convertInteger((String)h_Map.get("D_POLY"));
            d_str_st = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("STR_ST"));
            d_off_pro = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("OFF_PRO"));
            d_dis_km = FRGMLSTRK_Utility.convertDoubleArray((String)h_Map.get("DIS_KM"));
            d_ele_km = FRGMLSTRK_Utility.convertDoubleArray((String)h_Map.get("ELE_KM"));
            i_nob_di = FRGMLSTRK_Utility.convertInteger((String)h_Map.get("NOB_DI"));
            d_max_dep = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("MAX_DEP"));
            d_base_den = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("BASE_DEN"));
            input_area_name = FRGMLSTRK_Utility.convertString((String)h_Map.get("AREA_FE"))
            input_profile_num =
FRGMLSTRK_Utility.convertString((String)h_Map.get("NUM_PROFILE"));
            input_nob_gob =
FRGMLSTRK_Utility.convertDoubleArray((String)h_Map.get("OBS_ANO"));
            input_max_den = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("MAX_DEN"));
            input_min_den = FRGMLSTRK_Utility.convertDouble((String)h_Map.get("MIN_DEN"));
        }
        catch(Exception e) {

            e.printStackTrace();
        }
    }

    public void depthValues(){
        inter = 2;
        inter = inter + i_nob_di;
        d_dep_di_arr = new double[inter + 1];
        for (int i = 1 ;i <= inter; i++){

```

```
                d_dep_di_arr[i] =  FRGMLSTRK_PlotDepth.val2[i];

            }

        }

        public void cal() throws FRGMLSTRK_HandleException{

            getCoefficients();
            depthValues();

            d_den_for_arr = new double[inter];
            for (int i =1 ; i < inter; i++){
                d_den_for_arr[i] = FRGMLSTRK_PlotDensity.val3[i];

            }

            double z[] = new double[1000];
            ano = new double[i_no_obs + 1];
            x = new double[i_no_obs + 1];

            for (int i = 1; i <= i_no_obs; i++){

                x[i] = d_dis_km[i];

            }

            cftnt = d_cftnt_arr;
            int effd = 0;
            int jjk = 0;
            double []yy = new double[3];

            double []gg = new double[3];
            double []gs = new double[2500];
            double []gdmod = new double[2500];
            double GC;
            double []G = new double[i_no_obs+1];

            double zb;
            double wgc[][] = new double[2500][2500];
            for (int kk = 1; kk <= inter; kk++){

                if(kk == inter)
                    break;
                double zt = d_dep_di_arr[kk];
                zb = d_dep_di_arr[kk + 1];
                effd = effd + 1;
                yy[1] = d_str_st + d_off_pro;
                yy[2] = d_str_st - d_off_pro;
                double DX = ((x[2] - x[1]) / 10);
                double Zdif = zb - zt;
                int ND = (int)(Zdif / DX) + 1;
                int na1 = ND / 2;
                if ((ND - 2 * na1) < 0 || (ND - 2 * na1) > 0)
                    ND = ND + 1;

                double dz = Zdif / ND;
                int N2 = ND + 1;
                for (int JZ = 1; JZ <= N2; JZ++){
                    z[JZ] = zt + dz * (JZ - 1);
                }

                for (int k = 1; k <= i_no_obs; k++){

                    double xx = x[k];
                    for (int jz = 1; jz <= N2; jz++){

                        double DC;
                        DC = d_den_for_arr[effd] - d_base_den;
                        double sum = 0.0;
                        for (int jj = 1; jj <= i_d_poly + 1; jj++){

                            sum = sum + cftnt[jj] * Math.pow(z[jz], jj - 1);

                        }
```

```
                        for (int kl = 1; kl <= 2; kl++){

                            double effy = yy[kl];
                            double tr1 = Math.atan(effy/(z[jz]-d_ele_km[k]));
                            double ttp = Math.pow(-xx + sum, 2) + Math.pow(effy, 2) +
        Math.pow(z[jz] - d_ele_km[k], 2);
                            double tr2 = Math.atan((effy * (-xx + sum)) / ((z[jz] - d_ele_km[k]
        * Math.sqrt(ttp)));
                            gg[kl] = 13.3333 * DC * (tr1 - tr2);
                        }
                        gs[jz] = (gg[2] + gg[1]) / 2;
                    }
                    double DZ = z[2] - z[1];
                    double Sum1 = 0.0;
                    double Sum2 = 0.0;
                    double N1 = N2 / 2;
                    double N4 = N1 - 1;
                    for (int i3 = 1; i3 <= N1; i3++){

                        N2 = 2 * i3;
                        Sum1 = Sum1 + gs[N2];

                    }
                    for (int j2 = 1; j2 <= N4; j2++){

                        int N3 = 2 * j2 + 1;
                        Sum2 = Sum2 + gs[N3];
                    }
                    GC = gs[1] + 4 * Sum1 + 2 * Sum2 + gs[N2];
                    GC = (GC * DZ) / 3.0;
                    G[k] = GC;
                    jjk = jjk + 1;
                    gdmod[jjk] = G[k];

                }
                if (zb == d_max_dep){

                    for (int lk = 1;lk <= i_no_obs; lk++){

                        int klkl = lk;
                        for (int jh = 1; jh <= inter - 1; jh++){

                            wgc[lk][jh] = gdmod[klkl];
                            klkl = klkl + i_no_obs;
                        }
                    }
                }
            }

            for (int il = 1; il <= i_no_obs; il++){

                double sum = 0.0;
                for (int ih = 1; ih <= inter - 1; ih++){

                    sum = sum + wgc[il][ih];
                }
                ano[il] = sum;

            }
            double funct1=0;
            double err[]= new double [i_no_obs+1];
            for (int k = 1; k <= i_no_obs; k++) {

                err[k] = input_nob_gob[k] - ano[k];
                funct1 = funct1 + Math.pow( err[k] , 2 );
                //  System.out.println(err[k]);
            }
            setGraphValues(x, input_nob_gob, ano, cftnt);
        }


        public void getCoefficients() throws FRGMLSTRK_HandleException{

            double gmax = Math.abs(input_nob_gob[1]);
```

```
        for (int k = 1; k <= i_no_obs; k++) {

            if (Math.abs(input_nob_gob[k]) - gmax > 0)
                gmax = Math.abs(input_nob_gob[k]);

        }

        double datum = input_nob_gob[1];
        double r = input_nob_gob[i_no_obs] - input_nob_gob[1];
        int kk1 = 1;
        double gh = 0.5 * r;
        kk1 = kk1 + 1;
        double XH = 0;

        while ((( input_nob_gob[kk1] - datum) / gh ) - 1.0 < 0) {

            kk1 = kk1 + 1;
        }

        if ((( input_nob_gob[kk1] - datum) / gh) - 1.0 > 0) {

            XH = d_dis_km[kk1-1] + ( ( gh + datum - input_nob_gob[kk1-1]) * ( d_dis_km[kk1]
- d_dis_km[kk1-1] ) ) / ( input_nob_gob[kk1] - input_nob_gob[kk1-1] );
        }
        if ((( input_nob_gob[kk1] - datum ) / gh) - 1.0 == 0) {

            XH = d_dis_km[kk1];
        }

        double ini = XH;

        d_x_km_arr = new double[len + 1];
        d_z_km_arr = new double[len + 1];

        if (len <= i_d_poly){
            JFrame frame = null;
            JOptionPane.showMessageDialog(frame,
                    "The polynomial fit requires\n"
                    +"Order+1 data points.\n"
                    +"Use additional data points",
                    "Error!",
                    JOptionPane.ERROR_MESSAGE);
            throw new FRGMLSTRK_HandleException();
        }

        for (int i = 0; i <= len; i++){
            if ( i == 0){
                d_x_km_arr[i] = 0;

                d_z_km_arr[i] = 0;
            }
            if ( i == 1){
                d_x_km_arr[i] = ini;
                d_z_km_arr[i] = 0;
            }

            else{
                d_x_km_arr[i] = FRGMLSTRK_PlotFault.val[i];
                d_z_km_arr[i] = FRGMLSTRK_PlotFault.val1[i];
            }

        }


        Arrays.sort(d_x_km_arr);
        Arrays.sort(d_z_km_arr);
        double d_coeff_values[] = new double[i_d_poly + 1];
        double d_coeff_xzval_arr[] = new double[i_d_poly + 1];
        double d_coeff_zval_arr[][] = new double[i_d_poly + 1][i_d_poly + 1] ;

        double d_sum_lmatrix = 0;
        double d_sum_rmatrix = 0;

        for (int l = 0; l < i_d_poly + 1; l++){
```

```java

                d_sum_lmatrix = 0;
                for(int i = 0; i < this.d_x_km_arr.length; i++){

                    if (l == 0){

                        d_sum_lmatrix = d_sum_lmatrix + this.d_x_km_arr[i];
                    }
                    else{
                        double d_sum_xz = this.d_x_km_arr[i] * Math.pow(this.d_z_km_arr[i], l);
                        d_sum_lmatrix = d_sum_lmatrix+d_sum_xz;
                    }
                }

                d_coeff_xzval_arr[l] = d_sum_lmatrix;
            }

            double d_rmatrix_arr[] = new double[2 * i_d_poly + 1];
            for (int l = 0; l < d_rmatrix_arr.length; l++){

                d_sum_rmatrix = 0;
                for (int i = 0; i < d_x_km_arr.length; i++){

                    if (l == 0){

                        d_sum_rmatrix = d_sum_rmatrix + d_z_km_arr[i];
                        d_rmatrix_arr[l] = d_sum_rmatrix;
                    }
                    else{

                        double d_sum_zpow = Math.pow(d_z_km_arr[i], l);
                        d_sum_rmatrix = d_sum_rmatrix + d_sum_zpow;
                        d_rmatrix_arr[l] = d_sum_rmatrix;
                    }

                }

            }
            for (int i = 0; i < i_d_poly + 1; i++){

                for (int j = 0; j < i_d_poly + 1; j++){

                    if ( i == 0 && j == 0){

                        d_coeff_zval_arr[i][j] = d_x_km_arr.length - 1;

                    }
                    else{

                        d_coeff_zval_arr[i][j] = d_rmatrix_arr[i + j];
                    }

                }
            }

            int index[] = new int[i_d_poly + 1];
            d_coeff_values = solve(d_coeff_zval_arr, d_coeff_xzval_arr, index);


            d_cftnt_arr = new double[i_d_poly + 2];
            d_cftnt_arr[0] = 0.0;


            for (int i = 1; i <= i_d_poly + 1; i++){

                d_cftnt_arr[i] = d_coeff_values[i - 1];

            }

        }

        public  double[] solve(double a[][],
                double b[], int index[]) {
            int n = b.length;
            double x[] = new double[n];
```

```
            gaussian(a, index);

            for (int i = 0; i < n - 1; ++i) {
                for (int j = i + 1; j < n; ++j) {
                    b[index[j]] -= a[index[j]][i] * b[index[i]];

                }
            }


            x[n - 1] = b[index[n - 1]] / a[index[n - 1]][n - 1];
            for (int i = n - 2; i >= 0; --i) {
                x[i] = b[index[i]];
                for (int j = i + 1; j < n; ++j) {
                    x[i] -= a[index[i]][j] * x[j];
                }
                x[i] /= a[index[i]][i];

            }
            return x;
        }

        public  void gaussian(double a[][],
                int index[]) {
            int n = index.length;
            double c[] = new double[n];

            for (int i = 0; i < n; ++i) index[i] = i;

            for (int i = 0; i < n; ++i) {
                double c1 = 0;
                for (int j = 0; j < n; ++j) {
                    double c0 = Math.abs(a[i][j]);
                    if (c0 > c1) c1 = c0;
                }
                c[i] = c1;
            }

            int k = 0;
            for (int j = 0; j < n - 1; ++j) {
                double pi1 = 0;
                for (int i = j; i < n; ++i) {
                    double pi0 = Math.abs(a[index[i]][j]);
                    pi0 /= c[index[i]];
                    if (pi0 > pi1) {
                        pi1 = pi0;
                        k = i;
                    }
                }

                int itmp = index[j];
                index[j] = index[k];
                index[k] = itmp;
                for (int i = j + 1; i < n; ++i) {
                    double pj = a[index[i]][j] / a[index[j]][j];
                    a[index[i]][j] = pj;
                    for (int l=j+1; l<n; ++l)
                        a[index[i]][l] -= pj*a[index[j]][l];

                }
            }
        }


        public static void  setGraphValues(double []x,double []gobs, double []anomaly,double
    []coeff) {

            obj = new Object[i_no_obs + 1][3];
            DecimalFormat d1 = new DecimalFormat("0.#");
            DecimalFormat d = new DecimalFormat("0.##");
            DecimalFormat df = new DecimalFormat("0.###");
            DecimalFormat df1 = new DecimalFormat("0.####");
            for (int K = 1;K <= i_no_obs; K++){

                obj[K][0]= "" + d.format(x[K]);
```

```
            obj[K][1]= "" + df.format(gobs[K]);
            obj[K][2]= "" + df.format(anomaly[K]);

        }



        FRGMLSTRK_TableView.val.setText("");
        FRGMLSTRK_TableView.val.append("Coordinates of selected points on the fault plane(x,z):-\n");
        FRGMLSTRK_TableView.val.appendText("-------------------------------------------\n")
        for (int i = 1; i <= len; i++){

            FRGMLSTRK_TableView.val.appendText("("+d.format(d_x_km_arr[i])+","+d.format(d_z km_arr[i])+")\n") ;

        }
        FRGMLSTRK_TableView.val.appendText("\n");
        FRGMLSTRK_TableView.val.append("Polynomial coefficients of the fault plane:-\n");
        FRGMLSTRK_TableView.val.appendText("-------------------------------------------\n")
        FRGMLSTRK_TableView.val.appendText("\n");

        for (int i = 1; i < coeff.length; i++){
            FRGMLSTRK_TableView.val.appendText("f"+(i - 1)+" = " +df1.format(coeff[i])+ "\n");
        }

        FRGMLSTRK_TableView.val.appendText("\n");
        FRGMLSTRK_TableView.val.append("Depths of density interfaces:-\n");
        FRGMLSTRK_TableView.val.appendText("-------------------------------------------\n")
        FRGMLSTRK_TableView.val.appendText("\n");

        for (int i = 1;i <= FRGMLSTRK_CalculateValues.inter; i++){
            FRGMLSTRK_TableView.val.appendText("Density interface "+ i +" = " +d1.format(FRGMLSTRK_CalculateValues.d_dep_di_arr[i])+" (km)"+ "\n");
        }

        FRGMLSTRK_TableView.val.appendText("\n");
        FRGMLSTRK_TableView.val.append("Density of each formation :-\n");
        FRGMLSTRK_TableView.val.appendText("-------------------------------------------\n")
        FRGMLSTRK_TableView.val.appendText("\n");

        for (int i = 1; i < FRGMLSTRK_CalculateValues.inter; i++){
            FRGMLSTRK_TableView.val.appendText("Density value "+ i +" = " +d.format(FRGMLSTRK_CalculateValues.d_den_for_arr[i])+" (gm/cc)"+ "\n");
        }
    }
}
-------------------------------------------------------------------------------------------

package com.frgmlstrk.control;

import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.awt.*;
import java.io.*;
import java.text.DecimalFormat;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.frgmlstrk.model.FRGMLSTRK_CalculateValues;
import com.frgmlstrk.util.FRGMLSTRK_HandleException;
import com.frgmlstrk.view.FRGMLSTRK_MainPanel;
import com.frgmlstrk.view.graph.FRGMLSTRK_DensityGraph;
import com.frgmlstrk.view.graph.FRGMLSTRK_DepthLines;
import com.frgmlstrk.view.graph.FRGMLSTRK_PlainGraph;
import com.frgmlstrk.view.event.*;



public class FRGMLSTRK_Controller implements ActionListener {

    Object rowdata [][] = {};
    BufferedImage image;
    com.frgmlstrk.view.FRGMLSTRK_DrawGraph dg = new

```

```
com.frgmlstrk.view.FRGMLSTRK_DrawGraph();
    com.frgmlstrk.model.FRGMLSTRK_CalculateValues cv = new com.frgmlstrk.model.FRGMLSTRK_CalculateValues();
    public static boolean success = false;

    public void actionPerformed(ActionEvent ae)  {

        if (ae.getActionCommand().equals("Specify depth interfaces")) {

            try{
                if (FRGMLSTRK_CalculateValues.len <= FRGMLSTRK_CalculateValues.i_d_poly){
                    JFrame frame = null;
                    JOptionPane.showMessageDialog(frame,
                            "The polynomial fit requires\n"
                            +"degree+1 data points.\n"
                            +"Use additional data points",
                            "Error!",
                            JOptionPane.ERROR_MESSAGE);
                    throw new FRGMLSTRK_HandleException();
                }

                FRGMLSTRK_MainPanel.img2.removeMouseListener(FRGMLSTRK_PlotDepth.ml2);
                FRGMLSTRK_MainPanel.img2.removeMouseMotionListener((MouseMotionListener)FRGMLSTRK_PlotDepth.ml2);
                FRGMLSTRK_PlotDepth pf = new FRGMLSTRK_PlotDepth();
                cv.getAnamolyValues(FRGMLSTRK_MainPanel.captureValues());

                cv.depthValues();
                Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
                pf.paint(g);

            }
            catch(Exception e){

            }


        }else if(ae.getActionCommand().equals("Specify density values")){
            try{

                FRGMLSTRK_MainPanel.den.removeMouseListener(FRGMLSTRK_PlotDensity.ml4);
                FRGMLSTRK_MainPanel.den.removeMouseMotionListener((MouseMotionListener)FRGMLSTRK_PlotDensity.ml4);
                FRGMLSTRK_CalculateValues.inter = 2 + FRGMLSTRK_CalculateValues.i_nob_di;
                FRGMLSTRK_PlotDensity pf = new FRGMLSTRK_PlotDensity();
                Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
                pf.paint(g);
            }
            catch(Exception e) {

            }
        }else if(ae.getActionCommand().equals("Draw/Edit fault plane")){
            FRGMLSTRK_MainPanel.fl.removeMouseListener(FRGMLSTRK_EditFault.ml1);
            FRGMLSTRK_MainPanel.fl.removeMouseMotionListener((MouseMotionListener)FRGMLSTRK_EditFault.ml1);
            FRGMLSTRK_EditFault ef = new FRGMLSTRK_EditFault();
            try{
                if (FRGMLSTRK_CalculateValues.len <= FRGMLSTRK_CalculateValues.i_d_poly){
                    JFrame frame = null;
                    JOptionPane.showMessageDialog(frame,
                            "The polynomial fit requires\n"
                            +"degree+1 data points.\n"
                            +"Use additional data points",
                            "Error!",
                            JOptionPane.ERROR_MESSAGE);
                    throw new FRGMLSTRK_HandleException();
                }
                Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
                ef.paint(g);



            }
            catch(Exception e) {
                //e.printStackTrace();
```

```
            }
        }else if(ae.getActionCommand().equals("Load file")){

            FRGMLSTRK_MainPanel.loadData();
            FRGMLSTRK_MainPanel.p_Center.removeAll();
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.im);
            FRGMLSTRK_PlainGraph pg = new FRGMLSTRK_PlainGraph();
            try{
                if (FRGMLSTRK_CalculateValues.len <= FRGMLSTRK_CalculateValues.i_d_poly){
                    JFrame frame = null;
                    JOptionPane.showMessageDialog(frame,
                            "The polynomial fit requires\n"
                            +"degree+1 data points.\n"
                            +"Use additional data points",
                            "Error!",
                            JOptionPane.ERROR_MESSAGE);
                    throw new FRGMLSTRK_HandleException();
                }

                cv.getAnamolyValues(com.frgmlstrk.view.FRGMLSTRK_MainPanel.captureValues()
                cv.getCoefficients();
                cv.depthValues();
                Graphics2D g1 = (Graphics2D)FRGMLSTRK_MainPanel.im.getGraphics();
                pg.drawPlainGraph(g1);
                dg.plotZCoordinates(g1);
                pg.drawPlainGraph(g1);
                FRGMLSTRK_MainPanel.im.removeMouseListener(FRGMLSTRK_PlotFault.ml);
                FRGMLSTRK_MainPanel.im.removeMouseMotionListener((MouseMotionListener)FRGMI
TRK_PlotFault.ml);

            }
            catch(Exception e) {
                e.printStackTrace();
            }



        }

        else if(ae.getActionCommand().equals("Graph")){

            FRGMLSTRK_MainPanel.p_Center.removeAll();
            FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.img4);
            FRGMLSTRK_MainPanel.img4.setEditable(false);
            FRGMLSTRK_MainPanel.p_Center.validate();
            FRGMLSTRK_MainPanel.clearPanel(FRGMLSTRK_MainPanel.img4);
            try
            {
                int width = 1280;
                int height = 650;
                BufferedImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buffer.createGraphics();
                g1.setColor(Color.WHITE);
                g1.fillRect(0, 0, width, height);
                Graphics2D g2 = (Graphics2D)g1 ;
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dg.drawGraph(g2);
                dg.drawDepth(g2);
                dg.plotZCoordinates(g2);
                //dg.plotXYCoordinates(g2);
                dg.drawDen(g2);
                dg.plot(g2);
                dg.idex(g2);

                FileOutputStream os = new FileOutputStream(
FRGMLSTRK_CalculateValues.input_area_name +".jpg");
                ImageIO.write(buffer, "jpg", os);
                os.close();

                String path =  FRGMLSTRK_CalculateValues.input_area_name  +".jpg";
                image = ImageIO.read(new File(path));

```

```
                    Graphics g_image = FRGMLSTRK_MainPanel.img4.getGraphics();
                    g_image.drawImage(image, 0, 0,image.getWidth(), image.getHeight(), dg);

                    MouseListener ml3 = new MouseAdapter(){
                        public void mouseClicked(MouseEvent e){
                            Graphics g_image = FRGMLSTRK_MainPanel.img4.getGraphics();
                            g_image.drawImage(image, 0, 0,image.getWidth(), image.getHeight(),dg);
                        }
                    };
                    FRGMLSTRK_MainPanel.img4.addMouseListener(ml3);
                }
                catch (Exception e2) {

                    e2.printStackTrace();
                }


            }else if(ae.getActionCommand().equals("Save and Print")){

                try{

                    File img_file = new File(FRGMLSTRK_CalculateValues.input_area_name+".jpg");
                    JFileChooser saveFile = new JFileChooser();
                    File OutFile = saveFile.getSelectedFile();//new File(new File(INGRLSTRK_CalculateValues1.input_area_name+".html").getCanonicalPath()) ;  //saveFile.getSelectedFile();
                    FileWriter myWriter = null;

                    if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION){

                        OutFile = saveFile.getSelectedFile();

                        if (OutFile.canWrite() || !OutFile.exists()){

                            File dir = new File(OutFile.getParent());

                            success = img_file.renameTo(new File(dir,img_file.getName()));
                            myWriter = new FileWriter(OutFile+".html");
                            myWriter.write("<html> <Body onLoad = \"window.print()\"><table> <tr> <td>" +
                                    "<table border = 1> <tr> <th colspan = 4>LOCATION:-"+FRGMLSTRK_CalculateValues.input_area_name+"</th> </tr>");


                            DecimalFormat df =new DecimalFormat("0.###");
                            DecimalFormat d = new DecimalFormat("0.##");
                            DecimalFormat f1 = new DecimalFormat("0.#");
                            myWriter.write(" <tr><th colspan = 4> PROFILE NUMBER:-"+" "+FRGMLSTRK_CalculateValues.input_profile_num+" </th></tr>");
                            myWriter.write("<tr > <th>Distance (km) </th><th> Observed anamolie (mGal) </th> <th> Calculated anamolies (mGal) </th> </tr>");

                            for ( int K = 1; K <= FRGMLSTRK_CalculateValues.i_no_obs; K++){

                                myWriter.write("<tr> <td>" + d.format(FRGMLSTRK_CalculateValues.x[K])+"</td><td>"+df.format(FRGMLSTRK_CalculateValues.input_nob_gob[K])+"</td><td>"+df.format(FRGMLSTRK_CalculateValues.ano[K])+"</td></tr>");
                            }
                            myWriter.write(" </table> </td> <td> <img src = '"+ FRGMLSTRK_CalculateValues.input_area_name +".jpg'></td></tr></table><BR>Coordinates of selected points on the fault plane(x,z): <BR>");
                            myWriter.write("----------------------------------- <BR>");

                            DecimalFormat d1 =new DecimalFormat("0.####");

                            for (int i = 1; i <= FRGMLSTRK_CalculateValues.len; i++){

                                myWriter.write("("+d.format(FRGMLSTRK_CalculateValues.d_x_km_ar[i])+","+d.format(FRGMLSTRK_CalculateValues.d_z_km_arr[i])+")"+"<BR>") ;

                            }
```

```
                        myWriter.write("<BR>");
                        myWriter.write("Coefficient Values for the fault plane::"+"<BR>");
                        myWriter.write("----------------------------------------<BR>");
                        for ( int i = 1; i < FRGMLSTRK_CalculateValues.cftnt .length; i++)

                            myWriter.write("f"+ ( i - 1 ) +" = "+d1.format(FRGMLSTRK_CalculateValues.cftnt[i])+"<BR>");
                        }
                        myWriter.write("<BR>");
                        myWriter.write("Depths of density interfaces:"+"<BR>");
                        myWriter.write("----------------------------------- <BR>");
                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.inter; i++){
                            myWriter.write("Density interface "+i+" = "+f1.format(FRGMLSTRK_CalculateValues.d_dep_di_arr[i])+" (km)"+"<BR>");
                        }

                        myWriter.write("<BR>");
                        myWriter.write("Density of each formation:"+"<BR>");
                        myWriter.write("----------------------------------- <BR>");
                        for (int i = 1; i < FRGMLSTRK_CalculateValues.inter; i++){
                            myWriter.write("Density value "+i+" = "+d.format(FRGMLSTRK_CalculateValues.d_den_for_arr[i])+" (gm/cc)"+ "<BR>");
                        }
                        myWriter.close();
                    }
                }
                else
                {
                    //pops up error message

                }

            }
            catch(Exception e1) {

                e1.printStackTrace();
            }
        }
        else if (ae.getActionCommand().equals("Sample data")){

            FRGMLSTRK_MainPanel.setDefaultValues();
        }
        else if(ae.getActionCommand().equals("Save file")){

            try{


                JFileChooser saveFile = new JFileChooser();
                File OutFile = saveFile.getSelectedFile();//new File(new File(INGRLSTRK_CalculateValues1.input_area_name+".html").getCanonicalPath());  //saveFile.getSelectedFile();
                DecimalFormat d = new DecimalFormat("0.##");
                FileWriter myWriter = null;
                if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION){

                    OutFile = saveFile.getSelectedFile();

                    if (OutFile.canWrite() || !OutFile.exists()){

                        myWriter = new FileWriter(OutFile+".txt");
                        myWriter.write(""+FRGMLSTRK_CalculateValues.input_area_name);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.input_profile_num);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.d_str_st);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.i_nob_di);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.d_max_dep);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.d_base_den);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.i_no_obs);
```

```
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.d_off_pro);
                        myWriter.append(System.getProperty("line.separator"));

                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.i_no_obs; i++){

                            myWriter.write(""+(FRGMLSTRK_CalculateValues.d_dis_km[i])+",")

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.i_no_obs; i++){

                            myWriter.write(""+FRGMLSTRK_CalculateValues.d_ele_km[i]+",") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.i_no_obs; i++){

                            myWriter.write(""+(FRGMLSTRK_CalculateValues.input_nob_gob[i])-
,") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.i_d_poly);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.input_max_den);
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.input_min_den);
                        myWriter.append(System.getProperty("line.separator"));
                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.len; i++){

                            myWriter.write(""+d.format(FRGMLSTRK_CalculateValues.d_x_km_ar
i])+",") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));

                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.len; i++){

                            myWriter.write(""+d.format(FRGMLSTRK_CalculateValues.d_z_km_ar
i])+",") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.len);
                        myWriter.append(System.getProperty("line.separator"));
                        for (int i = 1; i <= FRGMLSTRK_CalculateValues.inter; i++){

                            myWriter.write(""+d.format(FRGMLSTRK_CalculateValues.d_dep_di_
r[i])+",") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        for (int i = 1; i < FRGMLSTRK_CalculateValues.inter; i++){

                            myWriter.write(""+d.format(FRGMLSTRK_CalculateValues.d_den_for_
rr[i])+",") ;

                        }
                        myWriter.append(System.getProperty("line.separator"));
                        myWriter.write(""+FRGMLSTRK_CalculateValues.inter);
                        myWriter.close();
                    }
                }
                else
                {
                    //pops up error message

                }

            }
            catch(Exception e1) {

                e1.printStackTrace();
            }
```

```java
            }

            else if (ae.getActionCommand().equals("Clear")){

                FRGMLSTRK_MainPanel.clearDefaultValues();
                FRGMLSTRK_MainPanel.p_Center.removeAll();
                com.frgmlstrk.view.FRGMLSTRK_TableView.populateEastPanel(rowdata);
                com.frgmlstrk.view.FRGMLSTRK_TableView.val.setText("");

            }else if(ae.getActionCommand().equals("Exit")){

                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit FGMLSTRK ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(success==false){
                        String fileName = FRGMLSTRK_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }else if(ae.getActionCommand().equals("Forward Modeling")){

                FRGMLSTRK_MainPanel.img3.removeMouseListener(FRGMLSTRK_EditDepth.ml1);
                FRGMLSTRK_MainPanel.img3.removeMouseMotionListener((MouseMotionListener)FRGMLSTRK_EditDepth.ml1);
                FRGMLSTRK_MainPanel.p_Center.removeAll();
                FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.graphLabel);
                FRGMLSTRK_MainPanel.p_Center.add(FRGMLSTRK_MainPanel.img3);
                FRGMLSTRK_MainPanel.img3.setEditable(false);
                FRGMLSTRK_MainPanel.p_Center.validate();
                cv.getAnamolyValues(FRGMLSTRK_MainPanel.captureValues());
                FRGMLSTRK_CalculateValues.inter = 2 + FRGMLSTRK_CalculateValues.i_nob_di;
                FRGMLSTRK_PlotDepth.val2[1] = 0;
                FRGMLSTRK_PlotDepth.val2[FRGMLSTRK_CalculateValues.inter] = FRGMLSTRK_CalculateValues.d_max_dep;

                try {
                    cv.cal();
                    com.frgmlstrk.view.FRGMLSTRK_TableView.populateEastPanel(FRGMLSTRK_CalculateValues.obj);
                    drawGraph(FRGMLSTRK_MainPanel.img3);
                } catch (Exception e) {

                }


                Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
                FRGMLSTRK_EditDensity eden = new FRGMLSTRK_EditDensity();
                eden.paint(g);
                FRGMLSTRK_EditDepth ed = new FRGMLSTRK_EditDepth();
                ed.paint(g);

                Graphics2D g2 = (Graphics2D)FRGMLSTRK_MainPanel.img3.getGraphics();
                g2.setFont(new Font("Arial", 40,20));
                g2.setColor(Color.black);
                g2.drawString("Number of coordinates "+FRGMLSTRK_CalculateValues.len, 710, 120);
                g2.drawLine(655, 0, 655, 85);
                g2.drawLine(655, 85, 990, 85);
                g2.drawString("Instructions",730, 20);
                g2.drawString("______________", 710, 20);
                g2.setFont(new Font("Arial", 40, 12));
                g2.setColor(Color.red);
                g2.drawString(" For modeling, press & drag the line segments of", 660, 40);
                g2.drawString(" depth/and density and release the mouse at",660,60);
                g2.drawString(" desired location", 660, 80);
                g2.setColor(Color.BLACK);


            }else if(ae.getActionCommand().equals("Specify fault coordinates")){

```

```
            cv.getAnamolyValues(FRGMLSTRK_MainPanel.captureValues());
            if(FRGMLSTRK_CalculateValues.d_dis_km.length - FRGMLSTRK_CalculateValues.i_no_obs != 1){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Distance values must be equal to\n"
                        +"number of observations.",
                        "Out of bounds error",
                        JOptionPane.ERROR_MESSAGE);
                try {

                    throw new FRGMLSTRK_HandleException();

                } catch (FRGMLSTRK_HandleException e) {

                }
            }
            if (FRGMLSTRK_CalculateValues.d_ele_km.length - FRGMLSTRK_CalculateValues.i_no_obs != 1){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Elevation values must be equal to\n"
                        +"number of observations.",
                        "Out of bounds error",
                        JOptionPane.ERROR_MESSAGE);
                try {

                    throw new FRGMLSTRK_HandleException();

                } catch (FRGMLSTRK_HandleException e) {

                }
            }

            if (FRGMLSTRK_CalculateValues.input_nob_gob.length - FRGMLSTRK_CalculateValues.i_no_obs != 1){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Observed anomalies must be equal to\n"
                        +"number of observations.",
                        "Out of bounds error",
                        JOptionPane.ERROR_MESSAGE);
                try {

                    throw new FRGMLSTRK_HandleException();

                } catch (FRGMLSTRK_HandleException e) {

                }
            }
            FRGMLSTRK_MainPanel.im.removeMouseListener(FRGMLSTRK_PlotFault.ml);
            FRGMLSTRK_MainPanel.im.removeMouseMotionListener((MouseMotionListener)FRGMLSTRK_PlotFault.ml);
            FRGMLSTRK_PlotFault pf = new FRGMLSTRK_PlotFault();
            Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
            pf.paint(g);

        }
    }

    public void drawGraph(TextArea t){

        FRGMLSTRK_DensityGraph de = new FRGMLSTRK_DensityGraph();
        FRGMLSTRK_DepthLines dl = new FRGMLSTRK_DepthLines();
        Graphics2D g2 = (Graphics2D)t.getGraphics();
        FRGMLSTRK_MainPanel.clearPanel(t);
        dg.plot(g2);
        dg.plotXYCoordinates(g2);
        dg.drawGraph(g2);
        dl.drawDepthLine(g2);
        dg.plotZCoordinates(g2);
        dg.plot(g2);
```

```
            de.Plainden(g2);
        }

    }
    --------------------------------------------------------------------------------------------

    package com.frgmlstrk.util;

    import javax.swing.JFrame;
    import javax.swing.JOptionPane;



    public class FRGMLSTRK_Utility {

        public static double convertDouble(String str) throws Exception {


            Double temp = null;

            try {
                temp = new Double(str.trim());

            }
            catch(Exception e){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Enter a numerical value.",
                        "Number format error",
                        JOptionPane.ERROR_MESSAGE);


            }
            return temp.doubleValue();
        }

        public static String convertString(String str) throws Exception {

            String temp = new String(str.trim());
            return temp;
        }

        public static int convertInteger(String str) throws Exception {


            Integer temp = null;
            try {
                temp = new Integer(str.trim());
            }
            catch(Exception e){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Enter a numerical value.",
                        "Number format error",
                        JOptionPane.ERROR_MESSAGE);

            }
            return temp.intValue();
        }

        public static int findMaximumNumber( double observe[]) {

            double max = 0.0d;
            for (int i = 0; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max = observe[i];
                }
            }

            int maxVal = (int) max/3*5;

```

```java
            return maxVal;
        }

        public static double findMinimumNumber( double observe[], double denVal) {

            double max = denVal;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) < Math.abs(max)) {

                    max = Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }

        public static double findMinimumNumber1( double observe[]) {

            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if ((observe[i]) < (max)) {

                    max = (observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }
        public static double findMaximumNumber1( double observe[]) {

            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max =Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }
        public static double findMaximumNumber( double observe[], double anoVal) {

            double max = anoVal;
            for (int i = 1; i < observe.length; i++) {

                if ((observe[i]) > (max)) {

                    max = (observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }



        public static double[] convertDoubleArray(String str) throws Exception {

            java.util.StringTokenizer st = new java.util.StringTokenizer(str, ",");
            String temp = "";
            java.util.ArrayList arr = new java.util.ArrayList();

            while(st.hasMoreTokens()) {

                temp = st.nextToken();
                arr.add(temp);
            }
            double d_array[] = new double[arr.size() + 1];
```

```java
            for (int i = 0; i <= arr.size(); i++) {

                if (i == 0)
                    d_array[i] = 0.0;
                else {

                    try {
                        d_array[i] = convertDouble( arr.get(i - 1).toString() );
                    }
                    catch(Exception e){
                        JFrame frame = null;
                        JOptionPane.showMessageDialog(frame,
                                "Enter numerical values.",
                                "Number format error",
                                JOptionPane.ERROR_MESSAGE);
                        throw new FRGMLSTRK_HandleException();

                    }
                }
            }
            return d_array;
        }
    }
```

```java
    package com.frgmlstrk.util;

    import java.awt.*;
    import com.frgmlstrk.view.FRGMLSTRK_MainPanel;

    public class FRGMLSTRK_HandleException extends Exception{

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public FRGMLSTRK_HandleException(){
            Graphics g = FRGMLSTRK_MainPanel.p_Center.getGraphics();
            g.setColor(Color.white);
            g.fillRect(0, 0, 1000, 600);
            g.setColor(Color.black);
            g.setFont(new Font("Arial", 20, 40));
            g.drawString("ERROR...", 400, 325);
        }
    }
```

```java
package com.gravlis.view;

import java.awt.Frame;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

import com.gravlis.control.GRAVLIS_Controller;
import com.gravlis.model.GRAVLIS_CalculateValues;




public class GRAVLIS_MainView extends Frame{
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static void main(String s[]){

        GRAVLIS_MainView cm = new GRAVLIS_MainView();
        cm.setSize(1280, 768);
        cm.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit GRAVLIS ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(GRAVLIS_Controller.success==false){
                        String fileName = GRAVLIS_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        });
        cm.setTitle("GRAVLIS");
        cm.setResizable(false);
        cm.add(new GRAVLIS_MainPanel());
        GRAVLIS_MainPanel.img.addMouseListener(new MouseAction());
        cm.setVisible(true);

    }

}

class MouseAction extends MouseAdapter{
    public void mousePressed(MouseEvent e) {

        GRAVLIS_CalculateValues.drawGraph();
    }

}

-------------------------------------------------------------------------------------------

package com.gravlis.view;

import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import javax.swing.JFileChooser;
import jxl.Cell;

```

```
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;


public class GRAVLIS_MainPanel extends Panel {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static TextArea img = new TextArea(46,140);
    Panel p_North, p_West;
    public static Panel p_East;

    static Panel p_South;

    public static Panel p_Center;
    static TextField inputValues [] = new TextField[16];

    Button actionButton[] = new Button[7];
    String rowdata[][]={};

    /**Field Area Name*/
    final static int AREA_FE = 0;
    /**Number of the Profile*/
    final static int NUM_PROFILE =  1;
    /**Number of observations*/
    final static int N_OBS = 2 ;
    /**Distance(km)*/
    final static int X_KM = 3;
    /**Elevation(km)*/
    final static int ELE_KM = 4;
    /**observed anomalies*/
    final static int NOB_GOB = 5;
    /**Y-values*/
    final static int Y_KM = 6;
    /** Surface density contrast (gm/cc) */
    final static int SD_POLY = 7;
    /**STRIKE-values*/
    final static int STRIKE_KM = 8;
    /**Maximum Depth*/
    final static int Z_VAL = 9;
    /**Number of formations*/
    final static int NUM_FOR =  10;
    /**Minimum Values*/
    final static int MIN_VAL =  11;
    /**Maximum values*/
    final static int MAX_VAL =  12;
    /**Number of iteration values*/
    final static int NUM_ITE =13;
    /**Density values*/
    final static int DEN_VAL =  14;
    /**Allowable Error*/
    final static int AL_ERR =   15;

    public GRAVLIS_MainPanel(){

        this.setLayout(new BorderLayout());
        p_North = new Panel();
        p_West = new Panel();
        p_East = new Panel ();
        p_South = new Panel();
        p_Center = new Panel();

        Label graphLabel = new Label("INVERSION OF GRAVITY ANOMALIES OF 2.5D LISTRIC FAULT
STRUCTURES USING ARBITRARY DENSITY DEPTH VARIATIONS", Label.CENTER);
        graphLabel.setFont(new Font("Bold", 1, 15));
        p_Center.add(graphLabel);

        for(int i = 0; i < 16; i++){
            inputValues[i] = new TextField();
        }
```

```java

            p_North.setFont(new Font("Bold",1,12));
            actionButton[0] = new Button("Load data");
            actionButton[1] = new Button("Interpretation with Fixed Depth");
            actionButton[2] = new Button("Interpretation with Fixed Density");
            actionButton[3] = new Button("Save & Print");
            actionButton[4] = new Button("Clear");
            actionButton[5] = new Button("Exit");

            this.populateNorthPanel();
            GRAVLIS_TableView.populateEastPanel(rowdata);

            this.add(p_North, BorderLayout.NORTH);
            p_Center.setSize(1200, 760);
            this.add(p_Center, BorderLayout.CENTER);
            img.setEditable(false);
            p_Center.add(img);
            this.add(p_East, BorderLayout.EAST);
            this.setVisible(true);
        }

        public void populateNorthPanel(){
            p_North.setLayout(new GridLayout(1,6));

            p_North.add(actionButton[0]);
            p_North.add(actionButton[1]);
            p_North.add(actionButton[2]);
            p_North.add(actionButton[3]);
            p_North.add(actionButton[4]);
            p_North.add(actionButton[5]);

            actionButton[0].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
            actionButton[1].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
            actionButton[2].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
            actionButton[3].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
            actionButton[4].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
            actionButton[5].addActionListener(new com.gravlis.control.GRAVLIS_Controller());
        }

        public static HashMap captureValues(){

            HashMap h_Map = new HashMap();

            try {
                h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
                h_Map.put("NUM_PROFILE", inputValues[NUM_PROFILE].getText());
                h_Map.put("N_OBS", inputValues[N_OBS].getText());
                h_Map.put("X_KM", inputValues[X_KM].getText());
                h_Map.put("ELE_KM", inputValues[ELE_KM].getText());
                h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
                h_Map.put("Y_KM", inputValues[Y_KM].getText());
                h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
                h_Map.put("STRIKE_KM", inputValues[STRIKE_KM].getText());
                h_Map.put("Z_VAL", inputValues[Z_VAL].getText());
                h_Map.put("NUM_FOR", inputValues[NUM_FOR].getText());
                h_Map.put("MIN_VAL", inputValues[MIN_VAL].getText());
                h_Map.put("MAX_VAL", inputValues[MAX_VAL].getText());
                h_Map.put("DEN_VAL", inputValues[DEN_VAL].getText());
                h_Map.put("NUM_ITE", inputValues[NUM_ITE].getText());
                h_Map.put("AL_ERR", inputValues[AL_ERR].getText());
            }
            catch (Exception e) {
                e.printStackTrace();
            }

            return h_Map;

        }

        public static void clearPanel(Panel p) {

            Graphics g = p.getGraphics();
            g.setColor(Color.WHITE);
            g.fillRect(0, 30, 1280, 650);
        }
```

```
    public static void loadData1()throws IOException  {
        try{

            String current = System.getProperty("user.dir");
            JFileChooser chooser=new  JFileChooser(current);
            int returnVal = chooser.showOpenDialog(null);
            String dis[], ele[],gobs[],min[],max[],den[],dep[];
            String disval = ""
,eleval="",gobsval="",minval="",maxval="",denval="",depval="";
            Workbook w;

            if(returnVal == JFileChooser.APPROVE_OPTION) {
                File f = chooser.getSelectedFile();
                w = Workbook.getWorkbook(f);
                Sheet sheet = w.getSheet(0);
                dis = new String[sheet.getRows()+1];
                ele = new String[sheet.getRows()+1];
                gobs = new String[sheet.getRows()+1];
                min = new String[sheet.getRows()+1];
                max = new String[sheet.getRows()+1];
                den = new String[sheet.getRows()+1];
                dep = new String[sheet.getRows()+1];
                for (int j = 0; j < sheet.getColumns(); j++) {
                    for (int i = 1; i < sheet.getRows(); i++) {
                        Cell cell = sheet.getCell(j, i);
                        CellType type = cell.getType();
                        if (type == CellType.LABEL) {
                            GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.AREA_FE].setTex
(cell.getContents());
                        }
                        if (type == CellType.NUMBER) {
                            if (j==1){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.NUM_PROFILE
.setText(cell.getContents());
                            }
                            if (j==2){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.N_OBS].setT
xt(cell.getContents());
                            }
                            if (j==3){
                                dis[i] = cell.getContents()+",";
                                disval = disval+dis[i];
                            }
                            if (j==4){
                                ele[i] = cell.getContents()+",";
                                eleval = eleval+ele[i];
                            }
                            if (j==5){
                                gobs[i] = cell.getContents()+",";
                                gobsval = gobsval+gobs[i];
                            }
                            if (j==6){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.STRIKE_KM].
etText(cell.getContents());
                            }
                            if (j==7){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.Y_KM].setTe
t(cell.getContents());
                            }
                            if (j==8){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.NUM_FOR].se
Text(cell.getContents());
                            }
                            if (j==9){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.SD_POLY].se
Text(cell.getContents());
                            }
                            if (j==10){
                                den[i] = cell.getContents()+",";
                                denval = denval+den[i];
                            }
                            if (j==11){
                                dep[i] = cell.getContents()+",";
```

```
                                depval = depval+dep[i];
                            }
                            if (j==12){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.NUM_ITE].setText(cell.getContents());
                            }
                            if (j==13){
                                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.AL_ERR].setText(cell.getContents());
                            }
                            try{
                                if (j==14){
                                    min[i] = cell.getContents()+",";

                                    minval = minval+min[i];
                                }
                                if (j==15){
                                    max[i] = cell.getContents()+",";

                                    maxval = maxval+max[i];
                                }
                            }
                            catch(Exception e){
                                //NullPointer Exception
                            }
                        }
                    }
                }
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.X_KM].setText(""+disval);

                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.ELE_KM].setText(""+eleval);
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.NOB_GOB].setText(""+gobsval);
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.MIN_VAL].setText(""+minval)
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.MAX_VAL].setText(""+maxval)
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.DEN_VAL].setText(""+denval)
                GRAVLIS_MainPanel.inputValues[GRAVLIS_MainPanel.Z_VAL].setText(""+depval);
            }
        }
        catch (BiffException e) {
            e.printStackTrace();
        }



    }

    public static void clearDefaultValues(){

        inputValues[AREA_FE].setText("");
        inputValues[NUM_PROFILE].setText("");
        inputValues[N_OBS].setText("");
        inputValues[X_KM].setText("");
        inputValues[ELE_KM].setText("");
        inputValues[NOB_GOB].setText("");
        inputValues[Y_KM].setText("");
        inputValues[SD_POLY].setText("");
        inputValues[STRIKE_KM].setText("");
        inputValues[Z_VAL].setText("");
        inputValues[NUM_FOR].setText("");
        inputValues[NUM_ITE].setText("");
        inputValues[AL_ERR].setText("");
    }
}
-------------------------------------------------------------------------------------------

package com.gravlis.view;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Panel;

```

```java
import java.awt.TextArea;

import javax.swing.JScrollPane;
import javax.swing.JTable;


public class GRAVLIS_TableView extends Panel{


    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static TextArea val = new TextArea(5,5);
    public static  void populateEastPanel(Object rowData[][]) {
        com.gravlis.view.GRAVLIS_MainPanel.p_East.removeAll();

        com.gravlis.view.GRAVLIS_MainPanel.p_East.setLayout(new GridLayout(2,1));


        Object columnNames[] = {"Distance(km)", "Observed anomalies (mGal)", "Calculated anomalies (mGal)", "Error (mGal)"};
        JTable table = new JTable(rowData, columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(300,500));
        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setAutoscrolls(true);

        com.gravlis.view.GRAVLIS_MainPanel.p_East.add(scrollPane);
        val.setEditable(false);
        com.gravlis.view.GRAVLIS_MainPanel.p_East.add(val);
        try{
            com.gravlis.view.GRAVLIS_MainPanel.p_East.validate();
        }
        catch(Exception e){

            Graphics g = GRAVLIS_MainPanel.img.getGraphics();
            g.setColor(Color.white);
            g.fillRect(0, 0, 1000, 600);
            g.setColor(Color.black);
            g.setFont(new Font("Arial", 20, 40));
            g.drawString("ERROR...", 300, 400);
        }
        com.gravlis.view.GRAVLIS_MainPanel.p_East.setVisible(true);


    }


}

-----------------------------------------------------------------------------------------------
```

```java
package com.gravlis.view;

import java.awt.*;
import java.applet.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.text.DecimalFormat;
import java.util.Arrays;

import com.gravlis.model.GRAVLIS_CalculateValues;
import com.gravlis.util.GRAVLIS_Utility;



public class GRAVLIS_DrawGraph extends Applet {


    /**
     *
     */
    private static final long serialVersionUID = 1L;
    public static int i_no_obs;

```

```java
        float maxY,maxZ,maxX ;
        public static float strike,strike1,strike2;
        double obs[];

        public void drawGraph(Graphics2D g2) {

            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(215-strike1, 50, 215-strike1, 300));
            String []a = {"A", "N", "O", "M", "A", "L", "Y", "(m", "G", "a", "l", "s)"};
            String []b = {"D", "E", "P", "T", "H", "(k", "m)"};
            for (int i = 0; i < a.length; i++) {
                g2.drawString(""+a[i], 100, 20 + 60 + ( i * 20 ) );
            }
            for (int i = 0; i < b.length; i++) {
                g2.drawString(""+b[i], 100, 20 + 350 + ( i * 20 ) );
            }

        }

        public void plot(Graphics2D g2) {

            g2.setFont( new Font("Arial", 12, 12) );
            DecimalFormat f = new DecimalFormat("0.#");
            g2.setColor(Color.BLACK);
            strike = (float) (GRAVLIS_CalculateValues.input_strike_km-GRAVLIS_CalculateValues.input_y_km);
            strike1 = (float) (65 * (strike / (2 * GRAVLIS_CalculateValues.input_strike_km)));
            strike2= (float) (40 * (strike /(2 * GRAVLIS_CalculateValues.input_strike_km)));
            i_no_obs = GRAVLIS_CalculateValues.input_n_obs;

            obs = new double[i_no_obs+1];
            for (int i = 1; i <= i_no_obs; i++) {
                obs[i] = GRAVLIS_CalculateValues.input_x_km[i];
            }

            maxX = (float)obs[i_no_obs];
            float maxy = (float) GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.input_nob_gob);
            float maxy1 = (float) GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.o_GC);
            if(maxy > maxy1)
                maxY = maxy;
            else
                maxY = maxy1;
            maxZ = (float)GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for];

            g2.drawString("|",(float) 600 + 65 - strike1, 260 + strike2);
            g2.drawString(""+f.format(GRAVLIS_CalculateValues.input_x_km[i_no_obs]),(float) 600 + 65 - strike1, 260 + strike2 - 8);
            g2.drawString("0", 125, 310);
            g2.drawString("DISTANCE(km)", 400, 250);
            float xplot = 0;
            float xInterval = (float) (GRAVLIS_CalculateValues.input_x_km[i_no_obs] / 5);

            int zInterval = 50;
            for (float x = xInterval, j = 1; x < 600; x += xInterval){
                xplot = xplot + xInterval;
                if(j > 4)
                    break;
                g2.drawString("|",(float) (215 + (450 * x / maxX) - strike1), 260 + strike2);
                g2.drawString("" + f.format(xplot), (float) (215 + (450 * x / maxX) - strike1) 3, 260 + strike2 - 8);
                j++;
            }

            DecimalFormat d =  new DecimalFormat("0.#");
            float points1 = maxZ / 5 ;
            for (int x = zInterval + 250,j = 1; x < 550; x += zInterval){
                g2.drawString("-", 148, 52 + x);
                g2.drawString("" +d.format(points1 * j), 125, 50 + x);
                j++;
            }

```

```java
    }


    public void plotXYCoordinates(Graphics2D g2){

        double minAno = GRAVLIS_Utility.findMinimumNumber1(GRAVLIS_CalculateValues.input_nob_gob);
        double maxAno = GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.input_nob_gob, minAno);
        double minObAno = GRAVLIS_Utility.findMinimumNumber1(GRAVLIS_CalculateValues.o_GC);
        double maxObAno = GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.o_GC, minObAno);

        if (minAno < 0 && maxObAno < 0 && maxAno < 0 && minObAno <0 ){
            plotXYCoordinates1(g2);
        }
        if (minAno >= 0 && maxObAno > 0){
            plotXYCoordinates1(g2);
        }
        if (minAno < 0 && maxObAno > 0 || maxAno>0 && minObAno<0){
            plotXYCoordinates2(g2);
        }
    }

    public void plotXYCoordinates1 (Graphics2D g2) {

        g2.setFont(new Font("Arial", 20, 12));
        g2.setColor(Color.black);
        float maxval = (float) GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.o_GC);
        float maxval1 = (float) GRAVLIS_Utility.findMaximumNumber(GRAVLIS_CalculateValues.input_nob_gob);
        if(Math.abs(maxval)>Math.abs(maxval1))
            maxY = maxval;
        else
            maxY = maxval1;
        int points = (int)maxY / 5;
        int yInterval = 50;
        g2.drawString("0", 215 - strike1 - 40, 50);
        for (int x = yInterval, j = 1; x < 250; x += yInterval){

            g2.drawString("-", 215 - strike1 - 2, 50 + x);
            g2.drawString("" + (points*j), 215 - strike1 - 40, 50 + x);
            j++;
        }
        float prevx = (float) (215 - strike1+( 450 * obs[1] / maxX));
        float prevy = (float)( 50 + ( 250 * GRAVLIS_CalculateValues.o_GC[1] / maxY ) );
        float xpoint = 0;
        float ypoint = 0;
        float gypoint = 0;

        for (int k = 1; k <= i_no_obs; k++) {

            xpoint = (float)( 450 * obs[k] / maxX);
            ypoint = (float)( ( 250 * GRAVLIS_CalculateValues.o_GC[k] / maxY ) );
            gypoint = (float)( ( 250 * GRAVLIS_CalculateValues.input_nob_gob[k] / maxY ) );

            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(prevx, prevy, 215-strike1+ xpoint, 50 + ypoint ));

            g2.setColor(Color.BLUE);
            g2.setFont(new Font("Arial", 20, 40));
            g2.drawString(".",  215-strike1+xpoint - 6 , 50 + gypoint + 3);

            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.black);
            prevx = 215-strike1 + xpoint;
            prevy = 50 + ypoint ;

        }

    }


    public void plotXYCoordinates2 (Graphics2D g2) {
```

```
        g2.setFont(new Font("Arial", 20, 12));
        g2.setColor(Color.black);
        int countPosObs=0,countNegObs=0,countPosCal=0,countNegCal=0;
        double store[] = new double[GRAVLIS_CalculateValues.input_n_obs+1];
        double store1[] = new double[GRAVLIS_CalculateValues.input_n_obs+1];
        double negStore[] = new double[GRAVLIS_CalculateValues.input_n_obs+1];
        double negStore1[] = new double[GRAVLIS_CalculateValues.input_n_obs+1];
        for(int i = 1; i <= GRAVLIS_CalculateValues.input_n_obs; i++){
            if(GRAVLIS_CalculateValues.o_GC[i]>0){
                store[i] = GRAVLIS_CalculateValues.o_GC[i];
                countPosCal = countPosCal+1;
            }
            else{
                negStore[i] = GRAVLIS_CalculateValues.o_GC[i];
                countNegCal = countNegCal+1;
            }
            if(GRAVLIS_CalculateValues.input_nob_gob[i]>0){
                store1[i] = GRAVLIS_CalculateValues.input_nob_gob[i];
                countPosObs = countPosObs+1;
            }
            else{
                negStore1[i] = GRAVLIS_CalculateValues.input_nob_gob[i];
                countNegObs = countNegObs+1;
            }
        }

        float maxPos = (float) GRAVLIS_Utility.findMaximumNumber1(store);
        float maxPos1 = (float) GRAVLIS_Utility.findMaximumNumber1(store1);
        float maxNeg = (float) GRAVLIS_Utility.findMaximumNumber(negStore);
        float maxNeg1 = (float) GRAVLIS_Utility.findMaximumNumber(negStore1);
        float posNum =0;

        if(maxPos>maxPos1)
            posNum = maxPos;
        else
            posNum = maxPos1;
        if(Math.abs(maxNeg)>Math.abs(maxNeg1))
            maxY = maxNeg;
        else
            maxY = maxNeg1;

        float prevx = (float) (215 - strike1+( 450 * obs[1] / maxX));;
        float prevy = 0;
        if(GRAVLIS_CalculateValues.o_GC[1]>0){
            if(countNegCal > countPosObs)
                prevy = 100 - (float)( ( 50 * GRAVLIS_CalculateValues.o_GC[1] / posNum ) );
            else
                prevy = 200 - (float)( ( 150 * GRAVLIS_CalculateValues.o_GC[1] / posNum ) )
        }
        else{
            if(countNegCal > countPosObs)
                prevy = 100 + (float)( ( 200 * GRAVLIS_CalculateValues.o_GC[1] / maxY ) );
            else{
                prevy = 200+(float)( ( 100 * GRAVLIS_CalculateValues.o_GC[1] / posNum ) );
            }
        }
        float xpoint = 0;
        float ypoint = 0;
        float gypoint = 0;
        float points = 0;

        DecimalFormat f = new DecimalFormat("0.#");
        if(countNegCal > countPosObs){
            g2.drawString("-", 215-strike1-4, 100);
            g2.drawString("0",215 - strike1 - 40,100);
        }
        else{
            g2.drawString("-", 215-strike1-4, 200);
            g2.drawString("0",215 - strike1 - 40,200);
        }
        g2.drawString("-", 215-strike1-4, 55);
        g2.drawString(""+f.format(posNum), 215-strike1-40, 55);

        if(countNegCal > countPosObs){
            points = maxY / 4;
```

```
            int yInterval=50;
            for (int x = yInterval, j = 1; x < 250; x+=yInterval){

                g2.drawString("-", 215 - strike1 - 4, 100 + x );
                g2.drawString("" + f.format(points * j), 215 - strike1 - 40, 100 + x );
                j++;
            }
        }
        else{
            points = posNum / 3;
            int yInterval=50;
            for (int x = yInterval, j = 1; x < 200; x+=yInterval){

                g2.drawString("-", 215 - strike1 - 4, 205 - x );
                g2.drawString("" + f.format(points * j), 215 - strike1 - 40, 205 - x );
                j++;
            }

            g2.drawString("-", 215 - strike1 - 4, 250 );
            g2.drawString("-" + f.format(posNum/2), 215 - strike1 - 40, 250 );
            g2.drawString("-", 215 - strike1 - 4, 300 );
            g2.drawString("-" + f.format(posNum), 215 - strike1 - 40, 300 );

        }
        for (int k = 1; k <= i_no_obs; k++) {

            xpoint = (float)( 450 * obs[k] / maxX);
            if(GRAVLIS_CalculateValues.o_GC[k]>0){
                if(countNegCal > countPosObs)
                    ypoint = 100-(float)( ( 50 * GRAVLIS_CalculateValues.o_GC[k] / posNum )
);
                else
                    ypoint = 200-(float)( ( 150 * GRAVLIS_CalculateValues.o_GC[k] / posNum
);
            }
            else{
                if(countNegCal > countPosObs)
                    ypoint = 100+(float)( ( 200 * GRAVLIS_CalculateValues.o_GC[k] / maxY )
);
                else
                    ypoint = 200+(float)( ( 100 * GRAVLIS_CalculateValues.o_GC[k] / posNum
);
            }
            if(GRAVLIS_CalculateValues.input_nob_gob[k]>0){
                if(countNegCal > countPosObs)
                    gypoint = 100-(float)( ( 50 * GRAVLIS_CalculateValues.input_nob_gob[k]
posNum ) );
                else
                    gypoint = 200-(float)( ( 150 * GRAVLIS_CalculateValues.input_nob_gob[k]
/ posNum ) );
            }
            else{
                if(countNegCal > countPosObs)
                    gypoint = 100+(float)( ( 200 * GRAVLIS_CalculateValues.input_nob_gob[k]
/ maxY ) );
                else
                    gypoint = 200+(float)( ( 100 * GRAVLIS_CalculateValues.input_nob_gob[k]
/ posNum ) );
            }
            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(prevx, prevy, 215-strike1 + xpoint,  ypoint  ));

            g2.setColor(Color.BLUE);
            g2.setFont(new Font("Arial", 20, 40));
            g2.drawString(".",  215-strike1+xpoint - 6 ,  gypoint+3 );

            g2.setFont(new Font("Arial", 20, 12));
            g2.setColor(Color.black);
            prevx = 215-strike1 + xpoint;
            prevy = ypoint ;

        }
    }


```

```
        public void drawDepth(Graphics2D g2) {

            DecimalFormat df = new DecimalFormat("0.#");
            float zpoint1 = 0;
            float zpoint = 0;
            if(GRAVLIS_CalculateValues.denval[1] <= 2.38)
                g2.setColor(Color.YELLOW);
            else
                g2.setColor(Color.GREEN);
            g2.fill(new Rectangle2D.Float(151, 300, 450, 250 ));
            g2.fillRect(150, 300,450, 250);
            for (float j = 300; j <= 550; j++){
                g2.draw(new Line2D.Float(600, j, 600+65, j-40 ));
            }
            Color col[] =
    {Color.BLACK,Color.ORANGE,Color.WHITE,Color.PINK,Color.YELLOW,Color.ORANGE,Color.PIN
    K,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
    olor.PINK,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
    olor.PINK,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
    olor.PINK,Color.WHITE,
                    Color.YELLOW,Color.ORANGE,Color.PINK,Color.WHITE,Color.YELLOW,Color.ORANGE,
    olor.PINK,Color.WHITE};

            Color col1[] =
    {Color.BLACK,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN,Color.D
    ARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
    Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
    Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
    Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,
                    Color.GREEN,Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA,Color.GREEN
    Color.DARK_GRAY,Color.BLUE,Color.CYAN,Color.MAGENTA};

            for (int i =1; i < GRAVLIS_CalculateValues.input_num_for; i++) {

                if (GRAVLIS_CalculateValues.denval[i+1] <= 2.38)
                    g2.setColor(col[i]);
                else
                    g2.setColor(col1[i]);
                zpoint = (float) (250 * GRAVLIS_CalculateValues.input_zval[i+1] / maxZ);
                zpoint1 = (float) (250 * GRAVLIS_CalculateValues.input_zval[i] / maxZ);
                if(i==1||i==GRAVLIS_CalculateValues.input_num_for-1){
                    if(GRAVLIS_CalculateValues.denval[i+1] <= 2.38)
                        g2.setColor(col[i]);
                    else{
                        g2.setColor(col1[i]);
                    }
                }
                g2.fill(new Rectangle2D.Float(151, 300 + zpoint1, 450, zpoint - zpoint1 ));
                for (float j = 300 + zpoint1;j <= 300 + zpoint; j++){
                    g2.draw(new Line2D.Float(600, j, 600 + 65, j - 40));
                }
                g2.setColor(Color.BLACK);
                g2.setFont(new Font("Arial", 40, 10));
                g2.drawString("----->" +
    df.format(GRAVLIS_CalculateValues.input_zval[i])+"(km)", 600 +65, 300 + zpoint1
    - 38);
            }
            zpoint1 = (float) (250 *
    GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for] / maxZ);
            g2.drawString("----->" +
    df.format(GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for])
    +"(km)", 600 +65, 300 + zpoint1 - 38);
            g2.setColor(Color.LIGHT_GRAY);
            for (float j = 151;j <= 600;j++){
                g2.draw(new Line2D.Float(j, 300, j + 65, 300 - 40));
            }
            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(215 - strike1, 260 + strike2, (float) 600 + 65 - strike1,
    260 + strike2));
```

```
        g2.draw(new Line2D.Float(600, 300, 600, 300+zpoint));
        g2.draw(new Line2D.Float(600 + 65, 300 - 40, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(600, 300 + zpoint, 600 + 65, 300 + zpoint - 40));
        g2.draw(new Line2D.Float(150, 550, 600, 550));
        g2.setColor(Color.BLACK);

        for (int i = 188; i <= 600 + 32; i++){
            g2.drawString("-", i, 280 + 4);
            i = i + 4;
        }
        g2.drawLine(90, 30, 950, 30);
        g2.drawLine(90, 565, 950, 565);
        g2.drawLine(950, 30, 950, 565);
        g2.drawLine(90, 30, 90, 565);
        g2.setFont(new Font("Arial", 40,12));
    }

    public void plotZCoordinates (Graphics2D g2) {
        i_no_obs = GRAVLIS_CalculateValues.input_n_obs;
        obs = new double[i_no_obs+1];
        for (int i = 1; i <= i_no_obs; i++) {
            obs[i] = GRAVLIS_CalculateValues.input_x_km[i];
        }
        maxX = (float) obs[i_no_obs];
        maxZ =
(float)GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for];
        float s = 0;
        float fc = 0;
        float spoint = 300;
        float fcpoint = (float)( 150 + ( 450 * GRAVLIS_CalculateValues.cft1[1] / maxX ) );
        float xpoint = 0;
        float zpoint = 0;

        while (s <=
GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for]) {

            float z1 = (float) 0.001;
            s = s + z1;
            fc = 0;
            for (int i = 1; i<4; i++){
                fc = (float) (fc + GRAVLIS_CalculateValues.cft1[i] * Math.pow(s, i - 1));
            }
            xpoint = (float)( 450 * fc / maxX);
            zpoint = (float)( 250 * s / maxZ);
            g2.setColor(Color.BLACK);
            g2.draw(new Line2D.Float(fcpoint, spoint,(float)150 + xpoint, 300 + zpoint));
            g2.setColor(Color.RED);
            g2.draw(new Line2D.Float(150, 300 + zpoint, (float) 150 + xpoint, 300 +
zpoint));
            fcpoint = (float) 150 + xpoint;
            spoint = 300 + zpoint;
            if (fcpoint > (float) 150 + ((700 * obs[i_no_obs] / maxX))){
                for (float j = spoint; j <= 550; j++){
                    g2.setColor(Color.RED);
                    g2.draw(new Line2D.Float(150, j, (float)( 150 + ((450 * obs[i_no_obs] /
maxX))), j));
                    g2.draw(new Line2D.Float((float)( 151 + ((450 * obs[i_no_obs] / maxX)))
j, (float)( 151 + ((450 * obs[i_no_obs] / maxX))) + 65, j - 40));
                }
                break;
            }
        }
    }


    public void drawDen(Graphics2D g2){
        g2.setFont(new Font("Arial", 20, 12));
        double minarr[]= new double[GRAVLIS_CalculateValues.denval.length];
        for (int i = 1; i <= GRAVLIS_CalculateValues.input_num_for; i++){
            minarr[i]= GRAVLIS_CalculateValues.denval[i];
        }
        Arrays.sort(minarr);
        int min_den = (int)minarr[1];
        denIndex(g2,min_den);
        double max_den = minarr[GRAVLIS_CalculateValues.input_num_for];
```

```
        double denMax1 = max_den - min_den;
        float denXpoint = 0;
        float denXpoint1 = 0;
        double denpoint1 = 0;
        float z1 = 0;
        float z2 = 0;
        float ini =0;
        double denpoint = GRAVLIS_CalculateValues.denval[1] - min_den ;
        denpoint1 = GRAVLIS_CalculateValues.denval[2] - min_den;
        denXpoint = (float) (90 * denpoint / denMax1);
        ini = (float) (250 * GRAVLIS_CalculateValues.input_zval[1] / maxZ);
        denXpoint1 = (float) (90 * denpoint1 / denMax1);
        g2.draw(new Line2D.Float(820 + denXpoint, 300 , 820 + denXpoint, 300+ini ));
        g2.draw(new Line2D.Float(820 + denXpoint, 300 + ini, 820 + denXpoint1, 300 + ini));
        for (int i = 1; i < GRAVLIS_CalculateValues.input_num_for; i++){
            denpoint = GRAVLIS_CalculateValues.denval[i+1] - min_den ;
            if (i < GRAVLIS_CalculateValues.input_num_for -1){
                denpoint1 = GRAVLIS_CalculateValues.denval[i + 2] - min_den;
                z1 = (float) (250 * GRAVLIS_CalculateValues.input_zval[i + 1] / maxZ);

            }
            denXpoint = (float) (90 * denpoint / denMax1);
            denXpoint1 = (float) (90 * denpoint1 / denMax1);
            z2  = (float) (250 * GRAVLIS_CalculateValues.input_zval[i] / maxZ);
            g2.setColor(Color.black);
            g2.draw(new Line2D.Float(820 + denXpoint, 300 + z2, 820 + denXpoint, 300 + z1));
            g2.draw(new Line2D.Float(820 + denXpoint, 300 + z1, 820 + denXpoint1, 300 +
z1));
        }
        g2.draw(new Line2D.Float(820 + denXpoint1, 300 + z1, 820 + denXpoint1, 550));
        g2.drawLine(820, 300, 910, 300);
        g2.drawLine(820, 300, 820, 550);
        g2.setColor(Color.white);
        g2.drawLine(821, 550, 910, 550);
    }

    public void denIndex(Graphics2D g,int minden){

        DecimalFormat d = new DecimalFormat("0.#");
        DecimalFormat d1 = new DecimalFormat("0.##");
        maxZ =
(float)GRAVLIS_CalculateValues.input_zval[GRAVLIS_CalculateValues.input_num_for];
        float points = maxZ / 5 ;
        int zInterval = 50;
        g.setColor(Color.black);
        for (int x = zInterval + 250, j = 1; x < 550; x += zInterval){
            g.drawString("-", 818, 52 + x);
            g.drawString("" +d.format(points * j), 795, 50 + x);
            j++;
        }
        g.setColor(Color.red);
        String []b = {"D", "E", "P", "T", "H", "(k", "m)"};
        for (int i = 0; i < b.length; i++) {
            g.drawString(""+b[i], 765,  350 + ( i * 20 ) );
        }
        g.setColor(Color.black);
        g.drawString(""+minden, 818, 298);
        double max = GRAVLIS_Utility.findMaximumNumber1(GRAVLIS_CalculateValues.denval);
        g.drawString(""+d1.format(max), 908, 298);
        g.setColor(Color.red);
        g.drawString("Density", 850, 280);
        g.drawString("(gm/cm  )",850 , 295);
        g.setFont(new Font("Arial", 20, 9));
        g.drawString("3",890 , 292);
        g.setFont(new Font("Arial", 20, 12));
        g.setColor(Color.black);
        g.drawString("0", 795, 305);

    }
    public void idex(Graphics2D g){
        g.setColor(Color.BLUE);
        g.setFont(new Font("Arial", 20, 50));
        g.drawString(" ...     ", 750, 67);
        g.setFont(new Font("Arial", 20, 12));
        g.drawString(": Observed anomalies", 820, 70);
```

```
            g.setColor(Color.BLACK);
            g.drawString("_____", 765, 85);
            g.drawString(":  Modeled anomalies", 820, 90);
        }

    }
    -----------------------------------------------------------------------------------------
```

```
    package com.gravlis.model;

    import java.awt.Color;
    import java.awt.Graphics;
    import java.awt.Graphics2D;
    import java.awt.image.BufferedImage;
    import java.io.File;
    import java.io.FileOutputStream;
    import java.text.DecimalFormat;
    import java.util.HashMap;

    import javax.imageio.ImageIO;
    import com.gravlis.model.GRAVLIS_CalNOREQ;
    import com.gravlis.model.GRAVLIS_CalculateValues;
    import com.gravlis.util.GRAVLIS_Utility;
    import com.gravlis.view.GRAVLIS_MainPanel;
    import com.gravlis.view.GRAVLIS_TableView;

    public class GRAVLIS_CalculateValues {

        public static Object obj[][] = null;
        public static double input_x_km[];public static double input_nob_gob[];
        public static double []o_GC ;public static double []o_err ;public static double
    []o_par;public static double []o_funct ;
        public static double []cft;public static double []cft1;public static double
    []denval;public static double []input_den_val;
        public static double []input_ele_km;public static double []input_min_val;public static
    double []input_max_val;
        public static double []par; public static double []var; public static double
    []input_zval;
        public static double input_y_km,input_sd_poly,input_al_err,o_func,input_strike_km;
        public static int o_iter,input_num_for,np,input_nob_iter,input_n_obs;
        public static int input_nregcft = 2;
        public static String input_area_name,input_profile_num="";
        public static String parValue = "";
        public static BufferedImage image;

        public void getAnamolyValues(HashMap h_Map)  {


            try {

                input_n_obs = GRAVLIS_Utility.convertInteger((String)h_Map.get("N_OBS"));
                input_x_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
                input_ele_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("ELE_KM"));
                input_nob_gob =
    GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                input_sd_poly = GRAVLIS_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                input_strike_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("STRIKE_KM"))
                input_y_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("Y_KM"));
                input_zval = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("Z_VAL"));
                input_num_for = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_FOR"));
                input_area_name = GRAVLIS_Utility.convertString((String)h_Map.get("AREA_FE"));
                input_profile_num =
    GRAVLIS_Utility.convertString((String)h_Map.get("NUM_PROFILE"));
                input_min_val =
    GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MIN_VAL"));
                input_max_val =
    GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MAX_VAL"));
                input_den_val =
    GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("DEN_VAL"));
                input_nob_iter = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_ITE"));
                input_al_err = GRAVLIS_Utility.convertDouble((String)h_Map.get("AL_ERR"));
            }
            catch(Exception e) {
                e.printStackTrace();

```

```
        }
        par = new double[input_nregcft + input_num_for + 2];
        var = new double[input_num_for + 1];
    }

    public void getFalDen(){


        o_GC  = new double[input_n_obs + 1];
        o_err = new double[input_n_obs + 1];
        o_par = new double[input_n_obs + 1];
        o_funct  = new double[input_nob_iter + 1];

        double []g1 = new double[input_n_obs + 1];
        double []g2 = new double[input_n_obs + 1];
        double gc[] = new double[input_n_obs + 1];
        double err[] = new double[input_n_obs + 1];

        double [][]p = new double[input_nregcft + input_num_for + 2][input_nregcft +
input_num_for + 3];
        double [][]s = new double[input_nregcft + input_num_for + 2][input_n_obs + 1];
        double []b = new double[input_nregcft + input_num_for + 2];
        double []par1 = new double[input_nregcft + input_num_for + 2];
        double []par2 = new double[input_nregcft + input_num_for + 2];
        double []dupar = new double[input_nregcft + input_num_for + 2];

        cft = new double[input_n_obs + 1];
        cft1 = new double[input_n_obs + 1];
        double []KS = new double[2];
        double funct2 = 0;
        double lambda = 0.5;
        np = input_nregcft + input_num_for + 1;
        denval = new double[input_num_for + 1];
        double funct1 =0;

        double datum = input_nob_gob[1];
        double r = input_nob_gob[input_n_obs]-input_nob_gob[1];
        int kk = 1;
        double gh = 0.5 * r;
        double XH = 0;
        while ((( input_nob_gob[kk] - datum) / gh ) - 1.0 < 0) {
            kk = kk + 1;
        }
        if ((( input_nob_gob[kk] - datum) / gh) - 1.0 > 0) {
            XH = input_x_km[kk-1] + ( ( gh + datum - input_nob_gob[kk-1]) * ( input_x_km[kk
- input_x_km[kk-1] ) ) / ( input_nob_gob[kk] - input_nob_gob[kk-1] );
        }
        if ((( input_nob_gob[kk] - datum ) / gh) - 1.0 == 0) {
            XH = input_x_km[kk];
        }
        cft[1] = XH;
        for(int k=1; k <= input_nregcft + 1; k++){
            par[k] = cft[k];
        }
        double dpar = 0.1;
        if(var == input_zval)
            GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np,
input_zval, gc);
        else
            GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np,
input_den_val, gc);
        for (int k = 1; k <= input_n_obs; k++) {
            err[k] = input_nob_gob[k] - gc[k];
            funct1 = funct1 + Math.pow( err[k] , 2 );
        }

        lambda = 0.5;
        int NP1 = np + 1;
        int IER = 1;

        while (IER <= input_nob_iter) {

            int ITER1 = IER ;
```

```
                o_funct[ITER1] = funct1;

                for (int K = 1; K <= np; K++) {
                    par1[K] = par[K];
                }

                for (int I = 1; I <= np; I++) {
                    par1[I] = par[I] + dpar / 2.0;
                    if(var == input_zval)
                        GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par1, np, input_zval, g1);
                    else
                        GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par1, np, input_den_val, g1);
                    par1[I] = par[I] - dpar / 2.0;
                    if(var == input_zval)
                        GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par1, np, input_zval, g2);
                    else
                        GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par1, np, input_den_val, g2);

                    for (int K = 1; K <= input_n_obs; K++) {
                        s[I][K] = ( g1[K] - g2[K] ) / dpar;
                    }
                }
                for (int J = 1; J <= NP1; J++) {
                    for(int I = 1; I <= np; I++) {
                        p[I][J] = 0.0;
                    }
                }
                for (int J = 1; J <= np; J++) {
                    for (int I = 1; I <= np; I++) {
                        for (int K = 1; K <= input_n_obs; K++){
                            p[I][J] = p[I][J] + s[I][K] * s[J][K];
                        }
                    }
                }

                for (int J = 1; J <= np; J++) {
                    for (int K = 1; K <= input_n_obs; K++) {
                        p[J][NP1] = p[J][NP1] + err[K] * s[J][K];
                    }
                }

                do {
                    double con = lambda + 1.0;
                    for (int I = 1; I <= np; I++) {
                        dupar[I] = par[I];
                    }
                    for (int L = 1; L <= np; L++) {
                        for (int J = 1;J <= np; J++) {
                            if (L - J == 0)
                                p[L][J] = p[L][J] * con;
                        }
                    }
                    GRAVLIS_CalNOREQ.getNOREQ(p, b, np, KS);
                    for (int I = 1; I <= np; I++) {
                        par2[I] = dupar[I] + 0.25 * b[I];
                    }
                    if(var==input_zval){
                        for(int j =1; j <= input_num_for; j++){
                            if (par2[input_nregcft + 1 + j] < input_min_val[j]) {
                                par2[input_nregcft +1 + j] = input_min_val[j];
                            }
                            if (par2[input_nregcft + 1 + j]>input_max_val[j]) {
                                par2[input_nregcft+ 1 + j] = input_max_val[j];
                            }
                        }
                    }
                    if(var == input_zval)
```

```
                    GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par2, np, input_zval, gc);
                else
                    GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par2, np, input_den_val, gc);
                funct2 = 0.0;
                for (int K = 1; K <= input_n_obs; K++) {
                    err[K] = input_nob_gob[K] - gc[K];
                    funct2 = funct2 + Math.pow(err[K] , 2);
                }
                if (funct1 - funct2 < 0) {

                    lambda = lambda * 2.0;
                    if (lambda - 12 < 0) {

                        for (int I = 1; I <= np; I++) {
                            for (int J = 1; J <= np; J++) {
                                if(I - J == 0)
                                    p[I][J] = p[I][J] / con;
                            }
                        }
                    }
                    else
                        break;

                }


            } while (funct1 - funct2 < 0);

            funct1 = funct2;
            DecimalFormat d = new DecimalFormat("0.#");
            IER++;
            for (int I = 1; I <= np; I++) {
                par[I] = par2[I];
            }
            o_iter = ITER1;
            o_func = funct2;
            parValue = d.format(par[1]);
            for (int K = 1;K <= input_n_obs; K++) {
                o_GC[K] = gc[K];
                o_err[K] = err[K];
            }
            for (int l = 1;l <= np; l++) {
                o_par[l] = par[l];
            }
            for(int j =1; j <= input_nregcft + 1; j++){
                cft1[j] = par[j];
            }
            if(var == input_zval){
                for(int j =1; j <= input_num_for; j++){
                    denval[j] = par[input_nregcft + 1 + j];
                }
            }
            else{
                for(int j = 1; j <= input_num_for; j++){
                    input_zval[j] = par[input_nregcft + 1 + j];
                    denval[j] = input_den_val[j];
                }
            }


            if ( funct2 < input_al_err || ITER1 == input_nob_iter||lambda -12 > 0 ) {

                o_iter = ITER1;
                o_func = funct2;
                parValue = d.format(par[1]);
                for (int K = 1;K <= input_n_obs; K++) {
                    o_GC[K] = gc[K];
                    o_err[K] = err[K];
                }
                for (int l = 1;l <= np; l++) {
                    o_par[l] = par[l];
```

```
                    }

                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC, o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }
                else{
                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC, o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }

                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                if ( funct2 < input_al_err || lambda -12 > 0)
                    break;

                lambda = lambda / 2.0;

            }
        }


        public static void  setGraphValues(int i_no_obs, int np, int ite, double []dis, double []GOBS, double []GCAL, double []ERROR, double []PARA, double FUNCT, String Area_fe) {

            obj = new Object[i_no_obs+1][4];

            DecimalFormat df = new DecimalFormat("0.###");
            DecimalFormat d = new DecimalFormat("0.##");
            DecimalFormat d1 = new DecimalFormat("0.#####");
            for(int K=1; K <= i_no_obs; K++){
                obj[K][0]= "" + dis[K];
                obj[K][1]= "" + df.format(GOBS[K]);
                obj[K][2]= "" + df.format(GCAL[K]);
                obj[K][3]= "" + df.format(ERROR[K]);
            }

            obj[0][0] ="ITERATION";
            obj[0][1] = "=" +" "+ite;

            GRAVLIS_TableView.val.setText("");
            GRAVLIS_TableView.val.appendText("ITERATION NUMBER:-"+ite);
            GRAVLIS_TableView.val.appendText("\n");
            GRAVLIS_TableView.val.appendText("OBJECTIVE FUNCTION  ="+d1.format(FUNCT)+"\n");
            GRAVLIS_TableView.val.appendText("\n");
            GRAVLIS_TableView.val.append("INTERPRETED PARAMETERS:-\n");
            GRAVLIS_TableView.val.appendText("-------------------------------------------\n");
            GRAVLIS_TableView.val.appendText("\n");
            GRAVLIS_TableView.val.appendText("COEFFICIENTS OF THE POLYNOMIAL ");
            GRAVLIS_TableView.val.appendText("\n");
            GRAVLIS_TableView.val.appendText("----------------------------------------------------\n");

            GRAVLIS_TableView.val.appendText(d1.format(PARA[1])+"\n");
            GRAVLIS_TableView.val.appendText(d1.format(PARA[2])+"\n");
            GRAVLIS_TableView.val.appendText(d1.format(PARA[3])+"\n");
            GRAVLIS_TableView.val.appendText("\n");

            if(var == input_zval)
                GRAVLIS_TableView.val.appendText("ESTIMATED DENSITIES OF THE FORMATIONS :-"+"\n");
            else
                GRAVLIS_TableView.val.appendText("ESTIMATED DEPTHS TO THE INTERFACES :-"+"\n");
            GRAVLIS_TableView.val.appendText("----------------------------------------------------------------\n");
            //GRAVLIS_TableView.val.appendText("\n");
            for(int k = 4; k <= input_nregcft + input_num_for + 1; k++){
                GRAVLIS_TableView.val.appendText(d.format(PARA[k])+"\n");
            }
```

```java
    }

    public static void drawGraph(){
        com.gravlis.view.GRAVLIS_DrawGraph dg = new com.gravlis.view.GRAVLIS_DrawGraph();
        try
        {
            int width = 1280;
            int height = 650;
            BufferedImage buffer = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
            Graphics g1= buffer.createGraphics();
            g1.setColor(Color.WHITE);
            g1.fillRect(0,0,width,height);
            Graphics2D g2 = (Graphics2D)g1 ;
            dg.plot(g2);
            dg.plotXYCoordinates(g2);
            dg.drawGraph(g2);
            dg.drawDepth(g2);
            dg.plotZCoordinates(g2);
            //dg.plotXYCoordinates(g2);
            dg.drawDen(g2);
            dg.plot(g2);
            dg.idex(g2);

            FileOutputStream os = new FileOutputStream( GRAVLIS_CalculateValues.input_area_name +".jpg");
            ImageIO.write(buffer, "jpg", os);
            os.close();

            String path =  GRAVLIS_CalculateValues.input_area_name  +".jpg";
            BufferedImage image = ImageIO.read(new File(path));

            Graphics g_image = GRAVLIS_MainPanel.img.getGraphics();
            g_image.drawImage(image, -40, 0, image.getWidth(), image.getHeight(), dg);


        }
        catch (Exception e2) {

        //  e2.printStackTrace();
        }
    }
}
package com.gravlis.model;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.text.DecimalFormat;
import java.util.HashMap;

import javax.imageio.ImageIO;
import com.gravlis.model.GRAVLIS_CalNOREQ;
import com.gravlis.model.GRAVLIS_CalculateValues;
import com.gravlis.util.GRAVLIS_Utility;
import com.gravlis.view.GRAVLIS_MainPanel;
import com.gravlis.view.GRAVLIS_TableView;

public class GRAVLIS_CalculateValues {

    public static Object obj[][] = null;
    public static double input_x_km[];public static double input_nob_gob[];
    public static double []o_GC ;public static double []o_err ;public static double []o_par;public static double []o_funct ;
    public static double []cft;public static double []cft1;public static double []denval;public static double []input_den_val;
    public static double []input_ele_km;public static double []input_min_val;public static double []input_max_val;
    public static double []par; public static double []var; public static double []input_zval;
    public static double input_y_km,input_sd_poly,input_al_err,o_func,input_strike_km;
    public static int o_iter,input_num_for,np,input_nob_iter,input_n_obs;
```

```java
        public static int input_nregcft = 2;
        public static String input_area_name,input_profile_num="";
        public static String parValue = "";
        public static BufferedImage image;

        public void getAnamolyValues(HashMap h_Map)  {


            try {

                input_n_obs = GRAVLIS_Utility.convertInteger((String)h_Map.get("N_OBS"));
                input_x_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
                input_ele_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("ELE_KM"));
                input_nob_gob = 
GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                input_sd_poly = GRAVLIS_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                input_strike_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("STRIKE_KM"));
                input_y_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("Y_KM"));
                input_zval = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("Z_VAL"));
                input_num_for = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_FOR"));
                input_area_name = GRAVLIS_Utility.convertString((String)h_Map.get("AREA_FE"));
                input_profile_num = 
GRAVLIS_Utility.convertString((String)h_Map.get("NUM_PROFILE"));
                input_min_val = 
GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MIN_VAL"));
                input_max_val = 
GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MAX_VAL"));
                input_den_val = 
GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("DEN_VAL"));
                input_nob_iter = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_ITE"));
                input_al_err = GRAVLIS_Utility.convertDouble((String)h_Map.get("AL_ERR"));
            }
            catch(Exception e) {
                e.printStackTrace();
            }
            par = new double[input_nregcft + input_num_for + 2];
            var = new double[input_num_for + 1];
        }

        public void getFalDen(){


            o_GC  = new double[input_n_obs + 1];
            o_err = new double[input_n_obs + 1];
            o_par = new double[input_n_obs + 1];
            o_funct  = new double[input_nob_iter + 1];

            double []g1 = new double[input_n_obs + 1];
            double []g2 = new double[input_n_obs + 1];
            double gc[] = new double[input_n_obs + 1];
            double err[] = new double[input_n_obs + 1];

            double [][]p = new double[input_nregcft + input_num_for + 2][input_nregcft + 
input_num_for + 3];
            double [][]s = new double[input_nregcft + input_num_for + 2][input_n_obs + 1];
            double []b = new double[input_nregcft + input_num_for + 2];
            double []par1 = new double[input_nregcft + input_num_for + 2];
            double []par2 = new double[input_nregcft + input_num_for + 2];
            double []dupar = new double[input_nregcft + input_num_for + 2];

            cft = new double[input_n_obs + 1];
            cft1 = new double[input_n_obs + 1];
            double []KS = new double[2];
            double funct2 = 0;
            double lambda = 0.5;
            np = input_nregcft + input_num_for + 1;
            denval = new double[input_num_for + 1];
            double funct1 =0;

            double datum = input_nob_gob[1];
            double r = input_nob_gob[input_n_obs]-input_nob_gob[1];
            int kk = 1;
            double gh = 0.5 * r;
            double XH = 0;
            while ((( input_nob_gob[kk] - datum) / gh ) - 1.0 < 0) {
```

```java
                kk = kk + 1;
            }
            if ((( input_nob_gob[kk] - datum) / gh) - 1.0 > 0) {
                XH = input_x_km[kk-1] + ( ( gh + datum - input_nob_gob[kk-1]) * ( input_x_km[kk
        - input_x_km[kk-1] ) ) / ( input_nob_gob[kk] - input_nob_gob[kk-1] );
            }
            if ((( input_nob_gob[kk] - datum ) / gh) - 1.0 == 0) {
                XH = input_x_km[kk];
            }
            cft[1] = XH;
            for(int k=1; k <= input_nregcft + 1; k++){
                par[k] = cft[k];
            }
            double dpar = 0.1;
            if(var == input_zval)
                GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np,
        input_zval, gc);
            else
                GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np,
        input_den_val, gc);
            for (int k = 1; k <= input_n_obs; k++) {
                err[k] = input_nob_gob[k] - gc[k];
                funct1 = funct1 + Math.pow( err[k] , 2 );
            }

            lambda = 0.5;
            int NP1 = np + 1;
            int IER = 1;

            while (IER <= input_nob_iter) {

                int ITER1 = IER ;
                o_funct[ITER1] = funct1;

                for (int K = 1; K <= np; K++) {
                    par1[K] = par[K];
                }

                for (int I = 1; I <= np; I++) {
                    par1[I] = par[I] + dpar / 2.0;
                    if(var == input_zval)
                        GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par1, np, input_zval, g1);
                    else
                        GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par1, np, input_den_val, g1);
                    par1[I] = par[I] - dpar / 2.0;
                    if(var == input_zval)
                        GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par1, np, input_zval, g2);
                    else
                        GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par1, np, input_den_val, g2);

                    for (int K = 1; K <= input_n_obs; K++) {
                        s[I][K] = ( g1[K] - g2[K] ) / dpar;
                    }
                }
                for (int J = 1; J <= NP1; J++) {
                    for(int I = 1; I <= np; I++) {
                        p[I][J] = 0.0;
                    }
                }
                for (int J = 1; J <= np; J++) {
                    for (int I = 1; I <= np; I++) {
                        for (int K = 1; K <= input_n_obs; K++){
                            p[I][J] = p[I][J] + s[I][K] * s[J][K];
                        }
                    }
```

```
                }

                for (int J = 1; J <= np; J++) {
                    for (int K = 1; K <= input_n_obs; K++) {
                        p[J][NP1] = p[J][NP1] + err[K] * s[J][K];
                    }
                }

                do {
                    double con = lambda + 1.0;
                    for (int I = 1; I <= np; I++) {
                        dupar[I] = par[I];
                    }
                    for (int L = 1; L <= np; L++) {
                        for (int J = 1;J <= np; J++) {
                            if (L - J == 0)
                                p[L][J] = p[L][J] * con;
                        }
                    }
                    GRAVLIS_CalNOREQ.getNOREQ(p, b, np, KS);
                    for (int I = 1; I <= np; I++) {
                        par2[I] = dupar[I] + 0.25 * b[I];
                    }
                    if(var==input_zval){
                        for(int j =1; j <= input_num_for; j++){
                            if (par2[input_nregcft + 1 + j] < input_min_val[j]) {
                                par2[input_nregcft +1 + j] = input_min_val[j];
                            }
                            if (par2[input_nregcft + 1 + j]>input_max_val[j]) {
                                par2[input_nregcft+ 1 + j] = input_max_val[j];
                            }
                        }
                    }
                    if(var == input_zval)
                        GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par2, np, input_zval, gc);
                    else
                        GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
        input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
        par2, np, input_den_val, gc);
                    funct2 = 0.0;
                    for (int K = 1; K <= input_n_obs; K++) {
                        err[K] = input_nob_gob[K] - gc[K];
                        funct2 = funct2 + Math.pow(err[K] , 2);
                    }
                    if (funct1 - funct2 < 0) {

                        lambda = lambda * 2.0;
                        if (lambda - 12 < 0) {

                            for (int I = 1; I <= np; I++) {
                                for (int J = 1; J <= np; J++) {
                                    if(I - J == 0)
                                        p[I][J] = p[I][J] / con;
                                }
                            }
                        }
                        else
                            break;

                    }


                } while (funct1 - funct2 < 0);

                funct1 = funct2;
                DecimalFormat d = new DecimalFormat("0.#");
                IER++;
                for (int I = 1; I <= np; I++) {
                    par[I] = par2[I];
                }
                o_iter = ITER1;
                o_func = funct2;
                parValue = d.format(par[1]);
```

```
                for (int K = 1;K <= input_n_obs; K++) {
                    o_GC[K] = gc[K];
                    o_err[K] = err[K];
                }
                for (int l = 1;l <= np; l++) {
                    o_par[l] = par[l];
                }
                for(int j =1; j <= input_nregcft + 1; j++){
                    cft1[j] = par[j];
                }
                if(var == input_zval){
                    for(int j =1; j <= input_num_for; j++){
                        denval[j] = par[input_nregcft + 1 + j];
                    }
                }
                else{
                    for(int j = 1; j <= input_num_for; j++){
                        input_zval[j] = par[input_nregcft + 1 + j];
                        denval[j] = input_den_val[j];
                    }
                }


                if ( funct2 < input_al_err || ITER1 == input_nob_iter||lambda -12 > 0 ) {

                    o_iter = ITER1;
                    o_func = funct2;
                    parValue = d.format(par[1]);
                    for (int K = 1;K <= input_n_obs; K++) {
                        o_GC[K] = gc[K];
                        o_err[K] = err[K];
                    }
                    for (int l = 1;l <= np; l++) {
                        o_par[l] = par[l];
                    }

                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC,
        o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }
                else{
                    setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC,
        o_err, o_par, o_func, input_area_name);
                    drawGraph();
                }

                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                if ( funct2 < input_al_err || lambda -12 > 0)
                    break;

                lambda = lambda / 2.0;

            }
        }


        public static void  setGraphValues(int i_no_obs, int np, int ite, double []dis, double
        []GOBS, double []GCAL, double []ERROR, double []PARA, double FUNCT, String Area_fe) {

            obj = new Object[i_no_obs+1][4];

            DecimalFormat df = new DecimalFormat("0.###");
            DecimalFormat d = new DecimalFormat("0.##");
            DecimalFormat d1 = new DecimalFormat("0.#####");
            for(int K=1; K <= i_no_obs; K++){
                obj[K][0]= "" + dis[K];
                obj[K][1]= "" + df.format(GOBS[K]);
                obj[K][2]= "" + df.format(GCAL[K]);
                obj[K][3]= "" + df.format(ERROR[K]);
```

```java
        }

        obj[0][0] ="ITERATION";
        obj[0][1] = "=" +" "+ite;

        GRAVLIS_TableView.val.setText("");
        GRAVLIS_TableView.val.appendText("ITERATION NUMBER:-"+ite);
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("OBJECTIVE FUNCTION  ="+d1.format(FUNCT)+"\n");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.append("INTERPRETED PARAMETERS:-\n");
        GRAVLIS_TableView.val.appendText("-------------------------------------------\n");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("COEFFICIENTS OF THE POLYNOMIAL ");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("------------------------------------------------------\n");

        GRAVLIS_TableView.val.appendText(d1.format(PARA[1])+"\n");
        GRAVLIS_TableView.val.appendText(d1.format(PARA[2])+"\n");
        GRAVLIS_TableView.val.appendText(d1.format(PARA[3])+"\n");
        GRAVLIS_TableView.val.appendText("\n");

        if(var == input_zval)
            GRAVLIS_TableView.val.appendText("ESTIMATED DENSITIES OF THE FORMATIONS :-"+"\n");
        else
            GRAVLIS_TableView.val.appendText("ESTIMATED DEPTHS TO THE INTERFACES :-"+"\n");
        GRAVLIS_TableView.val.appendText("------------------------------------------------------------------\n");
        //GRAVLIS_TableView.val.appendText("\n");
        for(int k = 4; k <= input_nregcft + input_num_for + 1; k++){
            GRAVLIS_TableView.val.appendText(d.format(PARA[k])+"\n");
        }
    }

    public static void drawGraph(){
        com.gravlis.view.GRAVLIS_DrawGraph dg = new com.gravlis.view.GRAVLIS_DrawGraph();
        try
        {
            int width = 1280;
            int height = 650;
            BufferedImage buffer = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
            Graphics g1= buffer.createGraphics();
            g1.setColor(Color.WHITE);
            g1.fillRect(0,0,width,height);
            Graphics2D g2 = (Graphics2D)g1 ;
            dg.plot(g2);
            dg.plotXYCoordinates(g2);
            dg.drawGraph(g2);
            dg.drawDepth(g2);
            dg.plotZCoordinates(g2);
            //dg.plotXYCoordinates(g2);
            dg.drawDen(g2);
            dg.plot(g2);
            dg.idex(g2);

            FileOutputStream os = new FileOutputStream( GRAVLIS_CalculateValues.input_area_name +".jpg");
            ImageIO.write(buffer, "jpg", os);
            os.close();

            String path =  GRAVLIS_CalculateValues.input_area_name  +".jpg";
            BufferedImage image = ImageIO.read(new File(path));

            Graphics g_image = GRAVLIS_MainPanel.img.getGraphics();
            g_image.drawImage(image, -40, 0, image.getWidth(), image.getHeight(), dg);


        }
        catch (Exception e2) {

        //  e2.printStackTrace();
        }
```

```java
        }
    }
    package com.gravlis.model;

    import java.awt.Color;
    import java.awt.Graphics;
    import java.awt.Graphics2D;
    import java.awt.image.BufferedImage;
    import java.io.File;
    import java.io.FileOutputStream;
    import java.text.DecimalFormat;
    import java.util.HashMap;

    import javax.imageio.ImageIO;
    import com.gravlis.model.GRAVLIS_CalNOREQ;
    import com.gravlis.model.GRAVLIS_CalculateValues;
    import com.gravlis.util.GRAVLIS_Utility;
    import com.gravlis.view.GRAVLIS_MainPanel;
    import com.gravlis.view.GRAVLIS_TableView;

    public class GRAVLIS_CalculateValues {

        public static Object obj[][] = null;
        public static double input_x_km[];public static double input_nob_gob[];
        public static double []o_GC ;public static double []o_err ;public static double []o_par;public static double []o_funct ;
        public static double []cft;public static double []cft1;public static double []denval;public static double []input_den_val;
        public static double []input_ele_km;public static double []input_min_val;public static double []input_max_val;
        public static double []par; public static double []var; public static double []input_zval;
        public static double input_y_km,input_sd_poly,input_al_err,o_func,input_strike_km;
        public static int o_iter,input_num_for,np,input_nob_iter,input_n_obs;
        public static int input_nregcft = 2;
        public static String input_area_name,input_profile_num="";
        public static String parValue = "";
        public static BufferedImage image;

        public void getAnamolyValues(HashMap h_Map)  {


            try {

                input_n_obs = GRAVLIS_Utility.convertInteger((String)h_Map.get("N_OBS"));
                input_x_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
                input_ele_km = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("ELE_KM"));
                input_nob_gob = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                input_sd_poly = GRAVLIS_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                input_strike_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("STRIKE_KM"));
                input_y_km = GRAVLIS_Utility.convertDouble((String)h_Map.get("Y_KM"));
                input_zval = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("Z_VAL"));
                input_num_for = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_FOR"));
                input_area_name = GRAVLIS_Utility.convertString((String)h_Map.get("AREA_FE"));
                input_profile_num = GRAVLIS_Utility.convertString((String)h_Map.get("NUM_PROFILE"));
                input_min_val = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MIN_VAL"));
                input_max_val = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("MAX_VAL"));
                input_den_val = GRAVLIS_Utility.convertDoubleArray((String)h_Map.get("DEN_VAL"));
                input_nob_iter = GRAVLIS_Utility.convertInteger((String)h_Map.get("NUM_ITE"));
                input_al_err = GRAVLIS_Utility.convertDouble((String)h_Map.get("AL_ERR"));
            }
            catch(Exception e) {
                e.printStackTrace();
            }
            par = new double[input_nregcft + input_num_for + 2];
            var = new double[input_num_for + 1];
        }

        public void getFalDen(){

```

```java

        o_GC  = new double[input_n_obs + 1];
        o_err = new double[input_n_obs + 1];
        o_par = new double[input_n_obs + 1];
        o_funct  = new double[input_nob_iter + 1];

        double []g1 = new double[input_n_obs + 1];
        double []g2 = new double[input_n_obs + 1];
        double gc[] = new double[input_n_obs + 1];
        double err[] = new double[input_n_obs + 1];

        double [][]p = new double[input_nregcft + input_num_for + 2][input_nregcft + input_num_for + 3];
        double [][]s = new double[input_nregcft + input_num_for + 2][input_n_obs + 1];
        double []b = new double[input_nregcft + input_num_for + 2];
        double []par1 = new double[input_nregcft + input_num_for + 2];
        double []par2 = new double[input_nregcft + input_num_for + 2];
        double []dupar = new double[input_nregcft + input_num_for + 2];

        cft = new double[input_n_obs + 1];
        cft1 = new double[input_n_obs + 1];
        double []KS = new double[2];
        double funct2 = 0;
        double lambda = 0.5;
        np = input_nregcft + input_num_for + 1;
        denval = new double[input_num_for + 1];
        double funct1 =0;

        double datum = input_nob_gob[1];
        double r = input_nob_gob[input_n_obs]-input_nob_gob[1];
        int kk = 1;
        double gh = 0.5 * r;
        double XH = 0;
        while ((( input_nob_gob[kk] - datum) / gh ) - 1.0 < 0) {
            kk = kk + 1;
        }
        if ((( input_nob_gob[kk] - datum) / gh) - 1.0 > 0) {
            XH = input_x_km[kk-1] + ( ( gh + datum - input_nob_gob[kk-1]) * ( input_x_km[kk] - input_x_km[kk-1] ) ) / ( input_nob_gob[kk] - input_nob_gob[kk-1] );
        }
        if ((( input_nob_gob[kk] - datum ) / gh) - 1.0 == 0) {
            XH = input_x_km[kk];
        }
        cft[1] = XH;
        for(int k=1; k <= input_nregcft + 1; k++){
            par[k] = cft[k];
        }
        double dpar = 0.1;
        if(var == input_zval)
            GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np, input_zval, gc);
        else
            GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for, input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km, par, np, input_den_val, gc);
        for (int k = 1; k <= input_n_obs; k++) {
            err[k] = input_nob_gob[k] - gc[k];
            funct1 = funct1 + Math.pow( err[k] , 2 );
        }

        lambda = 0.5;
        int NP1 = np + 1;
        int IER = 1;

        while (IER <= input_nob_iter) {

            int ITER1 = IER ;
            o_funct[ITER1] = funct1;

            for (int K = 1; K <= np; K++) {
                par1[K] = par[K];
            }

            for (int I = 1; I <= np; I++) {
```

```
                par1[I] = par[I] + dpar / 2.0;
                if(var == input_zval)
                    GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par1, np, input_zval, g1);
                else
                    GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par1, np, input_den_val, g1);
                par1[I] = par[I] - dpar / 2.0;
                if(var == input_zval)
                    GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par1, np, input_zval, g2);
                else
                    GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par1, np, input_den_val, g2);

                for (int K = 1; K <= input_n_obs; K++) {
                    s[I][K] = ( g1[K] - g2[K] ) / dpar;
                }
            }
            for (int J = 1; J <= NP1; J++) {
                for(int I = 1; I <= np; I++) {
                    p[I][J] = 0.0;
                }
            }
            for (int J = 1; J <= np; J++) {
                for (int I = 1; I <= np; I++) {
                    for (int K = 1; K <= input_n_obs; K++){
                        p[I][J] = p[I][J] + s[I][K] * s[J][K];
                    }
                }
            }

            for (int J = 1; J <= np; J++) {
                for (int K = 1; K <= input_n_obs; K++) {
                    p[J][NP1] = p[J][NP1] + err[K] * s[J][K];
                }
            }

            do {
                double con = lambda + 1.0;
                for (int I = 1; I <= np; I++) {
                    dupar[I] = par[I];
                }
                for (int L = 1; L <= np; L++) {
                    for (int J = 1;J <= np; J++) {
                        if (L - J == 0)
                            p[L][J] = p[L][J] * con;
                    }
                }
                GRAVLIS_CalNOREQ.getNOREQ(p, b, np, KS);
                for (int I = 1; I <= np; I++) {
                    par2[I] = dupar[I] + 0.25 * b[I];
                }
                if(var==input_zval){
                    for(int j =1; j <= input_num_for; j++){
                        if (par2[input_nregcft + 1 + j] < input_min_val[j]) {
                            par2[input_nregcft +1 + j] = input_min_val[j];
                        }
                        if (par2[input_nregcft + 1 + j]>input_max_val[j]) {
                            par2[input_nregcft+ 1 + j] = input_max_val[j];
                        }
                    }
                }
                if(var == input_zval)
                    GRAVLIS_CalNOREQ.getGF (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par2, np, input_zval, gc);
                else
                    GRAVLIS_CalNOREQ.getGF1 (input_n_obs, input_sd_poly, input_num_for,
input_nregcft, input_ele_km, input_strike_km, input_y_km, input_x_km,
par2, np, input_den_val, gc);
```

```
            funct2 = 0.0;
            for (int K = 1; K <= input_n_obs; K++) {
                err[K] = input_nob_gob[K] - gc[K];
                funct2 = funct2 + Math.pow(err[K] , 2);
            }
            if (funct1 - funct2 < 0) {

                lambda = lambda * 2.0;
                if (lambda - 12 < 0) {

                    for (int I = 1; I <= np; I++) {
                        for (int J = 1; J <= np; J++) {
                            if(I - J == 0)
                                p[I][J] = p[I][J] / con;
                        }
                    }
                }
                else
                    break;

            }


        } while (funct1 - funct2 < 0);

        funct1 = funct2;
        DecimalFormat d = new DecimalFormat("0.#");
        IER++;
        for (int I = 1; I <= np; I++) {
            par[I] = par2[I];
        }
        o_iter = ITER1;
        o_func = funct2;
        parValue = d.format(par[1]);
        for (int K = 1;K <= input_n_obs; K++) {
            o_GC[K] = gc[K];
            o_err[K] = err[K];
        }
        for (int l = 1;l <= np; l++) {
            o_par[l] = par[l];
        }
        for(int j =1; j <= input_nregcft + 1; j++){
            cft1[j] = par[j];
        }
        if(var == input_zval){
            for(int j =1; j <= input_num_for; j++){
                denval[j] = par[input_nregcft + 1 + j];
            }
        }
        else{
            for(int j = 1; j <= input_num_for; j++){
                input_zval[j] = par[input_nregcft + 1 + j];
                denval[j] = input_den_val[j];
            }
        }


        if ( funct2 < input_al_err || ITER1 == input_nob_iter||lambda -12 > 0 ) {

            o_iter = ITER1;
            o_func = funct2;
            parValue = d.format(par[1]);
            for (int K = 1;K <= input_n_obs; K++) {
                o_GC[K] = gc[K];
                o_err[K] = err[K];
            }
            for (int l = 1;l <= np; l++) {
                o_par[l] = par[l];
            }

            setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC, o_err, o_par, o_func, input_area_name);
            drawGraph();
        }
        else{
```

```java
                setGraphValues(input_n_obs, np, o_iter, input_x_km, input_nob_gob, o_GC, o_err, o_par, o_func, input_area_name);
                drawGraph();
            }

            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            if ( funct2 < input_al_err || lambda -12 > 0)
                break;

            lambda = lambda / 2.0;

        }
    }


    public static void  setGraphValues(int i_no_obs, int np, int ite, double []dis, double []GOBS, double []GCAL, double []ERROR, double []PARA, double FUNCT, String Area_fe) {

        obj = new Object[i_no_obs+1][4];

        DecimalFormat df = new DecimalFormat("0.###");
        DecimalFormat d = new DecimalFormat("0.##");
        DecimalFormat d1 = new DecimalFormat("0.#####");
        for(int K=1; K <= i_no_obs; K++){
            obj[K][0]= "" + dis[K];
            obj[K][1]= "" + df.format(GOBS[K]);
            obj[K][2]= "" + df.format(GCAL[K]);
            obj[K][3]= "" + df.format(ERROR[K]);
        }

        obj[0][0] ="ITERATION";
        obj[0][1] = "=" +" "+ite;

        GRAVLIS_TableView.val.setText("");
        GRAVLIS_TableView.val.appendText("ITERATION NUMBER:-"+ite);
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("OBJECTIVE FUNCTION  ="+d1.format(FUNCT)+"\n");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.append("INTERPRETED PARAMETERS:-\n");
        GRAVLIS_TableView.val.appendText("-------------------------------------------\n");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("COEFFICIENTS OF THE POLYNOMIAL ");
        GRAVLIS_TableView.val.appendText("\n");
        GRAVLIS_TableView.val.appendText("----------------------------------------------------\n");

        GRAVLIS_TableView.val.appendText(d1.format(PARA[1])+"\n");
        GRAVLIS_TableView.val.appendText(d1.format(PARA[2])+"\n");
        GRAVLIS_TableView.val.appendText(d1.format(PARA[3])+"\n");
        GRAVLIS_TableView.val.appendText("\n");

        if(var == input_zval)
            GRAVLIS_TableView.val.appendText("ESTIMATED DENSITIES OF THE FORMATIONS :-"+"\n");
        else
            GRAVLIS_TableView.val.appendText("ESTIMATED DEPTHS TO THE INTERFACES :-"+"\n");
        GRAVLIS_TableView.val.appendText("----------------------------------------------------------------\n");
        //GRAVLIS_TableView.val.appendText("\n");
        for(int k = 4; k <= input_nregcft + input_num_for + 1; k++){
            GRAVLIS_TableView.val.appendText(d.format(PARA[k])+"\n");
        }
    }

    public static void drawGraph(){
        com.gravlis.view.GRAVLIS_DrawGraph dg = new com.gravlis.view.GRAVLIS_DrawGraph();
        try
        {
            int width = 1280;
```

```java
                int height = 650;
                BufferedImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buffer.createGraphics();
                g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);
                Graphics2D g2 = (Graphics2D)g1 ;
                dg.plot(g2);
                dg.plotXYCoordinates(g2);
                dg.drawGraph(g2);
                dg.drawDepth(g2);
                dg.plotZCoordinates(g2);
                //dg.plotXYCoordinates(g2);
                dg.drawDen(g2);
                dg.plot(g2);
                dg.idex(g2);

                FileOutputStream os = new FileOutputStream(
GRAVLIS_CalculateValues.input_area_name +".jpg");
                ImageIO.write(buffer, "jpg", os);
                os.close();

                String path =  GRAVLIS_CalculateValues.input_area_name  +".jpg";
                BufferedImage image = ImageIO.read(new File(path));

                Graphics g_image = GRAVLIS_MainPanel.img.getGraphics();
                g_image.drawImage(image, -40, 0, image.getWidth(), image.getHeight(), dg);


            }
            catch (Exception e2) {

            //  e2.printStackTrace();
            }
        }
}
---------------------------------------------------------------------------------------------

package com.gravlis.model;

public class GRAVLIS_CalNOREQ {

    public static double []vsd = null;
    public static double []dep = null;
    public static int N2=0;

    public static void main(String[] args) {

        //Methods that support the main class
    }

    public static double []getGF(int n,double bden,int nfm,int ndgre,double ele[],double
strk,double offset,double []x,double par[],double np,double []zr,double []ano) {

        double []cft = new double[ndgre + 2];
        double []den = new double[nfm + 2];
        for(int k = 1; k <= ndgre + 1; k++){
            cft[k] = par[k];
        }
        for(int k = 1; k <= nfm; k++){
            den[k] = par[ndgre + 1 + k];
        }
        int effd = 0;
        int jjk = 0;
        int kk=1;
        double zt =0;
        double []Z ;
        double []YY = new double[3];
        double []GS ;
        double []GG = new double[3];
        double wgc[][] = new double[2500][2500];
        double []gmod = new double[2500];
        double []gdmod = new double[2500];
        double GC = 0;

```

```java
        do{
            effd = effd + 1;
            YY[1] = strk + offset;
            YY[2] = strk - offset;
            double DX = ( x[2] - x[1] ) / 10;
            double ZB = zr[kk];
            double zdif = ZB - zt;
            int NDIV = (int)( zdif / DX ) + 1;
            int N1 = NDIV / 2;
            if (NDIV - (2 * N1 ) < 0 || NDIV - ( 2 * N1 ) > 0) {
                NDIV = NDIV + 1;
            }
            double DZ = zdif / NDIV;
            N2 = NDIV + 1;
            Z = new double[N2+1];
            GS = new double[N2+1];
            vsd = new double[N2+1];
            dep = new double[N2+1];
            for (int JZ = 1; JZ <= N2; JZ++) {
                Z[JZ] = zt + DZ * ( JZ - 1 );
            }

            double dc = den[effd] - bden;
            for (int K = 1; K <= n; K++) {
                double XX = x[K] ;

                for ( int JZ = 1;JZ <= N2; JZ++) {
                    double sum = 0;
                    for(int jj = 1; jj <= ndgre + 1; jj++){
                        sum = sum + cft[jj] * Math.pow(Z[JZ], jj - 1);
                    }
                    vsd[JZ] = dc;
                    dep[JZ] = Z[JZ];

                    for (int KL = 1; KL <= 2; KL++) {
                        double EFFY = YY[KL];
                        double tr1 = Math.atan( EFFY / ( Z[JZ] - ele[K] ) );
                        double ttp = Math.pow( ( -XX + sum) , 2) + Math.pow( EFFY , 2) + Math.pow( ( Z[JZ] - ele[K] ) , 2 );
                        double tr2 = Math.atan( ( EFFY * ( -XX + sum) ) / ( ( Z[JZ] - ele[K] ) * Math.sqrt( ttp ) ) );
                        GG[KL] = 13.3333 * dc * ( tr1 - tr2 );

                    }
                    GS[JZ] = ( ( GG[2] + GG[1] ) / 2);
                }
                gmod[K]= getSIMP(GS,Z,N2,GC);
                jjk = jjk + 1;
                gdmod[jjk] = gmod[K];
            }
            kk = kk + 1;
            if(kk - nfm <= 0)
                zt = zr[kk - 1];
        }while(kk - nfm <= 0);
        for (int lk = 1; lk <= n; lk++){
            int klkl = lk;
            for (int jh = 1; jh <= nfm; jh++){
                wgc[lk][jh] = gdmod[klkl];
                klkl = klkl + n;
            }
        }

        for (int il = 1; il <= n; il++){
            double sum = 0.0;
            for (int ih = 1; ih <= nfm; ih++){
                sum = sum + wgc[il][ih];
            }
            ano[il] = sum;
        }
        return ano;
    }
    public static double []getGF1(int n,double bden,int nfm,int ndgre,double ele[],double strk,double offset,double []x,double par[],double np,double []den,double []ano) {

```

```
            double []cft = new double[ndgre + 2];
            double []dep = new double[nfm + 5];
            for(int k = 1; k <= ndgre + 1; k++){
                cft[k] = par[k];
            }
            for(int k=1; k <= nfm; k++){
                dep[k] = par[ndgre + 1 + k];
            }
            int effd = 0;
            int jjk = 0;
            int kk = 1;
            double zt =0;
            double []Z ;
            double []YY = new double[3];
            double []GS ;
            double []GG = new double[3];
            double wgc[][] = new double[2500][2500];
            double []gmod = new double[2500];
            double []gdmod = new double[2500];
            double GC = 0;

            do{
                effd = effd+1;
                YY[1] = strk + offset;
                YY[2] = strk - offset;
                double DX = ( x[2] - x[1] ) / 10;
                double ZB = dep[kk];
                double zdif = ZB - zt;
                int NDIV = (int)( zdif / DX ) + 1;
                int N1 = NDIV / 2;
                if (NDIV - (2 * N1 ) < 0 || NDIV - ( 2 * N1 ) > 0) {
                    NDIV = NDIV + 1;
                }
                double DZ = zdif / NDIV;
                N2 = NDIV + 1;
                Z = new double[N2 + 1];
                GS = new double[N2 + 1];
                vsd = new double[N2 + 1];
                for (int JZ = 1; JZ <= N2; JZ++) {
                    Z[JZ] = zt + DZ * ( JZ - 1 );
                }
                double dc = den[effd] - bden;
                for (int K = 1; K <= n; K++) {
                    double XX = x[K] ;
                    for ( int JZ = 1; JZ <= N2; JZ++) {
                        double sum = 0;
                        for(int jj = 1; jj <= ndgre + 1; jj++){
                            sum = sum + cft[jj] * Math.pow(Z[JZ], jj - 1);
                        }
                        vsd[JZ] = dc;

                        for (int KL = 1; KL <= 2; KL++) {
                            double EFFY = YY[KL];
                            double tr1 = Math.atan( EFFY / ( Z[JZ] - ele[K] ) );
                            double ttp = Math.pow( ( -XX + sum) , 2) + Math.pow( EFFY , 2) + Math.pow( ( Z[JZ] - ele[K] ) , 2 );
                            double tr2 = Math.atan( ( EFFY * ( -XX + sum) ) / ( ( Z[JZ] - ele[K] ) * Math.sqrt( ttp ) ) );
                            GG[KL] = 13.3333 * dc * ( tr1 - tr2 );
                        }
                        GS[JZ] = ( ( GG[2] + GG[1] ) / 2);
                    }
                    gmod[K] = getSIMP(GS,Z,N2,GC);
                    jjk = jjk + 1;
                    gdmod[jjk] = gmod[K];
                }
                kk = kk + 1;
                if(kk - nfm <= 0)
                    zt = dep[kk - 1];
            }while(kk - nfm <= 0);
            for (int lk = 1; lk <= n; lk++){
                int klkl = lk;
                for (int jh = 1; jh <= nfm; jh++){
                    wgc[lk][jh] = gdmod[klkl];
                    klkl = klkl + n;
```

```
            }
        }


        for (int il = 1; il <= n; il++){
            double sum = 0.0;
            for (int ih = 1; ih <= nfm; ih++){
                sum = sum + wgc[il][ih];
            }
            ano[il] = sum;
        }
        return ano;
    }
    public static double getSIMP(double []gs,double []z,int n,double ggc) {

        double dz = z[2] - z[1];
        double sum1 = 0.0;
        double sum2 = 0.0;
        int n1 = n / 2;
        int n4 = n1 - 1;
        for(int I = 1; I <= n1; I++) {
            int n2 = 2 * I;
            sum1 = sum1 + gs[n2];
        }
        for(int I = 1; I <= n4; I++) {
            int n3 = 2 * I +1;
            sum2 = sum2 + gs[n3];
        }
        ggc = gs[1] + 4 * sum1 + 2 * sum2 + gs[n];
        ggc = ggc * dz / 3.0;
        return ggc;
    }
    public static double []getNOREQ(double p[][], double b[], int n, double KS[]) {

        int I = n + 1;
        double []a = new double[n * n + 1];
        for (int I1 = 1; I1 <= n; I1++) {
            for (int I2 = 1; I2 <= n; I2++) {
                int I3 = (I1 - 1) * n + I2;
                a[I3] = p[I2][I1];
            }
        }

        for (int I4 = 1;I4 <= n; I4++) {
            b[I4] = p[I4][I];
        }
        double TOL = 0;
        KS[0] = 0;
        int JJ = - n;
        int IT;
        int NY = 0;
        for (int J = 1; J <= n; J++) {
            int JY = J + 1;
            JJ = JJ + n + 1;
            double biga = 0;
            IT = JJ - J;
            int imax = 0;
            for (int i = J; i <= n; i++) {
                int IJ = IT + i;
                if (Math.abs(biga) - Math.abs(a[IJ]) < 0) {
                    biga = a[IJ];
                    imax = i;
                }
            }
            int I1 = 0;
            if (Math.abs(biga) - TOL <= 0) {
                KS[1] = 1;
                return KS;
            }
            else {
                I1 = J + n * (J - 2);
                IT = imax - J;
            }
            double save;
            for (int K = J; K <= n; K++) {
```

```java
                I1 = I1 + n;
                int I2 = I1 + IT;
                save = a[I1];
                a[I1] = a[I2];
                a[I2] = save;
                a[I1] = a[I1] / biga;
            }

            save = b[imax];
            b[imax] = b[J];
            b[J] = save / biga;
            int IQS = 0;

            if (J - n < 0 || J - n > 0) {
                IQS = n * (J - 1);
                for (int IX = JY; IX <= n; IX++) {
                    int IXJ = IQS + IX;
                    IT = J - IX;
                    for (int JX = JY; JX <= n; JX++) {
                        int IXJX = n * (JX - 1) + IX;
                        int JJX = IXJX + IT;
                        a[IXJX] = a[IXJX] - (a[IXJ] * a[JJX]);
                    }
                    b[IX] = b[IX] - (b[J] * a[IXJ]);
                }
            }
        }
        NY = n - 1;
        IT = n * n;
        for (int J = 1; J <= NY; J++) {
            int ia = IT - J;
            int ib = n - J;
            int ic = n;
            for (int K = 1; K <= J; K++) {
                b[ib] = b[ib] - a[ia] * b[ic];
                ia = ia - n;
                ic = ic - 1;
            }
        }
        return b;
    }
}
```

```
------------------------------------------------------------------------------------------
```

```java
package com.gravlis.control;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;
import java.io.*;
import java.text.DecimalFormat;
import javax.swing.*;



import com.gravlis.model.GRAVLIS_CalculateValues;
import com.gravlis.view.GRAVLIS_MainPanel;
import com.gravlis.view.GRAVLIS_TableView;


public class GRAVLIS_Controller implements ActionListener{

    String rowdata[][]={};
    com.gravlis.model.GRAVLIS_CalculateValues cv = new
com.gravlis.model.GRAVLIS_CalculateValues();
    FileWriter myWriter = null;
    public static boolean success = false;

    public void actionPerformed(ActionEvent ae) {

        if(ae.getActionCommand().equals("Interpretation with Fixed Depth")) {

            com.gravlis.view.GRAVLIS_TableView.populateEastPanel(rowdata);

```

```
                GRAVLIS_TableView.val.setText("");

                cv.getAnamolyValues(com.gravlis.view.GRAVLIS_MainPanel.captureValues());
                for(int k=1;k<=GRAVLIS_CalculateValues.input_num_for;k++){
                    GRAVLIS_CalculateValues.par[k+GRAVLIS_CalculateValues.input_nregcft+1] =
    GRAVLIS_CalculateValues.input_den_val[k];
                }
                GRAVLIS_CalculateValues.var = GRAVLIS_CalculateValues.input_zval;

                cv.getFalDen();
                com.gravlis.view.GRAVLIS_TableView.populateEastPanel(GRAVLIS_CalculateValues.ob
    );
                GRAVLIS_CalculateValues.obj = null;

                com.gravlis.view.GRAVLIS_MainPanel.p_East.repaint();
                com.gravlis.view.GRAVLIS_MainView mv = new com.gravlis.view.GRAVLIS_MainView();
                mv.setResizable(true);

            }else if(ae.getActionCommand().equals("Interpretation with Fixed Density")){
                com.gravlis.view.GRAVLIS_TableView.populateEastPanel(rowdata);
                GRAVLIS_TableView.val.setText("");

                cv.getAnamolyValues(com.gravlis.view.GRAVLIS_MainPanel.captureValues());
                for(int k=1;k<=GRAVLIS_CalculateValues.input_num_for;k++){
                    GRAVLIS_CalculateValues.par[k+GRAVLIS_CalculateValues.input_nregcft+1] =
    GRAVLIS_CalculateValues.input_zval[k];
                }
                GRAVLIS_CalculateValues.var = GRAVLIS_CalculateValues.input_den_val;

                cv.getFalDen();

                com.gravlis.view.GRAVLIS_TableView.populateEastPanel(GRAVLIS_CalculateValues.ob
    );
                GRAVLIS_CalculateValues.obj = null;

                com.gravlis.view.GRAVLIS_MainPanel.p_East.repaint();
                com.gravlis.view.GRAVLIS_MainView mv = new com.gravlis.view.GRAVLIS_MainView();
                mv.setResizable(true);
            }
            else if(ae.getActionCommand().equals("Save & Print")){

                try{
                    String current = System.getProperty("user.dir");
                    File img_file = new File( GRAVLIS_CalculateValues.input_area_name+".jpg");
                    JFileChooser saveFile = new JFileChooser(current);
                    File OutFile = saveFile.getSelectedFile();
                    if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
                    {
                        OutFile = saveFile.getSelectedFile();

                        if (OutFile.canWrite() || !OutFile.exists())
                        {
                            File dir = new File(OutFile.getParent());
                            GRAVLIS_Controller.success = img_file.renameTo(new
    File(dir,img_file.getName()));
                            //System.out.println("save successful" + success);
                            myWriter = new FileWriter(OutFile+".html");
                            myWriter.write(" </table> </td> <td> <img src = '"+
    GRAVLIS_CalculateValues.input_area_name
    +".jpg'></td></tr></table>");
                            myWriter.write("<html><Body onLoad = \"window.print()\"> <table>
    <tr> <td>" +
                                    "<table border = 1> <tr> <th colspan = 4>LOCATION:-
    "+GRAVLIS_CalculateValues.input_area_name+"</th> </tr>");


                            DecimalFormat df =new DecimalFormat("0.###");

                            myWriter.write(" <tr><th colspan = 4> PROFILE NUMBER:-"+"
    "+GRAVLIS_CalculateValues.input_profile_num+" </th></tr>");
                            myWriter.write(" <tr><th colspan = 4> ITERATION"+"
    "+GRAVLIS_CalculateValues.o_iter+" </th></tr>");

                            myWriter.write("<tr > <th>Distance (km) </th> <th> Observed
    anomalies (mGal) </th> <th> Calculated anomalies (mGal) </th> <th>
```

```
Error (mGal) </th></tr>");

                        for ( int K = 1; K <= GRAVLIS_CalculateValues.input_n_obs; K++){

                            myWriter.write("<tr> <td>" +
GRAVLIS_CalculateValues.input_x_km[K]+"</td>
<td>"+df.format(GRAVLIS_CalculateValues.input_nob_gob[K])+"</td>
<td>"+df.format(GRAVLIS_CalculateValues.o_GC[K])+"</td>
<td>"+df.format(GRAVLIS_CalculateValues.o_err[K])+"</td></tr>");
                        }
                        myWriter.write("</table>");

                        DecimalFormat d =new DecimalFormat("0.##");
                        DecimalFormat d1 =new DecimalFormat("0.#####");

                        myWriter.write("<BR>");
                        myWriter.write("OBJECTIVE FUNCTION ="+"
"+d1.format(GRAVLIS_CalculateValues.o_func)+"<BR>");
                        myWriter.write("<BR>");
                        myWriter.write("INTERPRETED PARAMETERS:"+"<BR>");
                        myWriter.write("----------------------------------- <BR>");
                        myWriter.write("<BR>");
                        myWriter.write(" COEFFICIENTS OF THE POLYNOMIAL :-"+"<BR>");
                        myWriter.write("---------------------------------------------
<BR>");
                        myWriter.write(d1.format(GRAVLIS_CalculateValues.o_par[1])+"<BR>");
                        myWriter.write(d1.format(GRAVLIS_CalculateValues.o_par[2])+"<BR>");
                        myWriter.write(d1.format(GRAVLIS_CalculateValues.o_par[3])+"<BR>");
                        myWriter.write("<BR>");
                        if(GRAVLIS_CalculateValues.var==GRAVLIS_CalculateValues.input_zval)
                            myWriter.write("ESTIMATED DENSITIES OF THE FORMATIONS
:-"+"<BR>");
                        else
                            myWriter.write("ESTIMATED DEPTHS TO THE INTERFACES :-"+"<BR>");
                        myWriter.write("------------------------------------------------------
---- <BR>");
                        for ( int i = 4; i <=
GRAVLIS_CalculateValues.input_nregcft+GRAVLIS_CalculateValues.input_
num_for+1; i++) {
                            myWriter.write(d.format(GRAVLIS_CalculateValues.o_par[i])+"<BR>"
);
                        }
                        myWriter.close();
                    }
                }
                else
                {
                    //pops up error message

                }

            }
            catch(Exception e1) {
                e1.printStackTrace();
            }

        }else if(ae.getActionCommand().equals("Load data")){
            try {
                GRAVLIS_MainPanel.loadData1();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }else if(ae.getActionCommand().equals("Clear")){
            GRAVLIS_MainPanel.clearDefaultValues();
            clearStaticValues();
            com.gravlis.view.GRAVLIS_MainPanel.clearPanel(GRAVLIS_MainPanel.p_Center);
            com.gravlis.view.GRAVLIS_TableView.populateEastPanel(rowdata);
            GRAVLIS_TableView.val.setText("");
            Graphics g = GRAVLIS_MainPanel.img.getGraphics();
            g.setColor(Color.white);
            g.fillRect(0, 0, 1000, 600);

        }else if(ae.getActionCommand().equals("Exit")){
            JFrame frame = null;
```

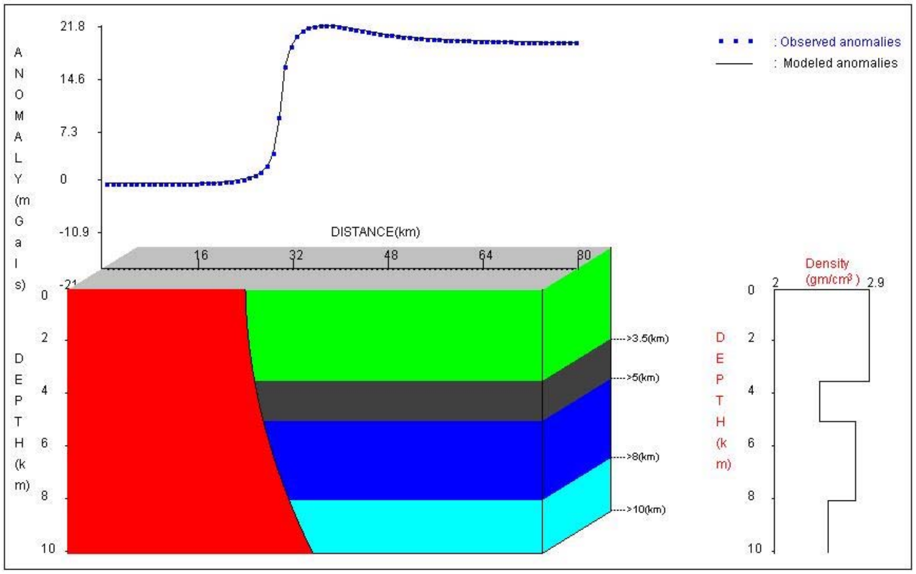
```
            int r = JOptionPane.showConfirmDialog(
                    frame,
                    "Exit GRAVLIS ?",
                    "Confirm Exit ",
                    JOptionPane.YES_NO_OPTION);
            if(r == JOptionPane.YES_OPTION ){
                if(success==false){
                    String fileName = GRAVLIS_CalculateValues.input_area_name+".jpg";
                    File f = new File(fileName);
                    f.delete();
                }
                System.exit(0);
            }

        }
    }
    public void clearStaticValues(){
        GRAVLIS_CalculateValues.cft = null;
        GRAVLIS_CalculateValues.cft1 = null;
        GRAVLIS_CalculateValues.denval = null;
        GRAVLIS_CalculateValues.input_al_err = 0;
        GRAVLIS_CalculateValues.input_area_name = null;
        GRAVLIS_CalculateValues.input_den_val = null;
        GRAVLIS_CalculateValues.input_ele_km = null;
        GRAVLIS_CalculateValues.input_max_val = null;
        GRAVLIS_CalculateValues.input_min_val = null;
        GRAVLIS_CalculateValues.input_n_obs = 0;
        GRAVLIS_CalculateValues.input_nob_gob = null;
        GRAVLIS_CalculateValues.input_nob_iter = 0;
        GRAVLIS_CalculateValues.input_num_for = 0;
        GRAVLIS_CalculateValues.input_sd_poly = 0;
        GRAVLIS_CalculateValues.input_strike_km = 0;
        GRAVLIS_CalculateValues.input_x_km = null;
        GRAVLIS_CalculateValues.input_y_km = 0;
        GRAVLIS_CalculateValues.input_zval = null;
        GRAVLIS_CalculateValues.o_err = null;
        GRAVLIS_CalculateValues.o_funct = null;
        GRAVLIS_CalculateValues.o_GC = null;
        GRAVLIS_CalculateValues.o_iter = 0;
        GRAVLIS_CalculateValues.o_par = null;
        System.gc();

    }
}
```

```
package com.gravlis.util;


import javax.swing.JFrame;
import javax.swing.JOptionPane;



public class GRAVLIS_Utility {

    public static double convertDouble(String str) throws Exception {


        Double temp = null;

        try {
            temp = new Double(str.trim());

        }
        catch(Exception e){

            JFrame frame = null;
            JOptionPane.showMessageDialog(frame,
                    "Enter a numerical value.",
                    "Number format error",
                    JOptionPane.ERROR_MESSAGE);
```

```
            }
            return temp.doubleValue();
        }

        public static String convertString(String str) throws Exception {

            String temp = new String(str.trim());
            return temp;
        }

        public static int convertInteger(String str) throws Exception {


            Integer temp = null;
            try {
                temp = new Integer(str.trim());
            }
            catch(Exception e){

                JFrame frame = null;
                JOptionPane.showMessageDialog(frame,
                        "Enter a numerical value.",
                        "Number format error",
                        JOptionPane.ERROR_MESSAGE);

            }
            return temp.intValue();
        }

        public static double findMaximumNumber( double observe[]) {

            double max = 0.0d;
            for (int i = 0; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max = observe[i];
                }
            }

            double maxVal =  max/3*5;
            return maxVal;
        }

        public static double findMinimumNumber( double observe[], double denVal) {

            double max = denVal;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) < Math.abs(max)) {

                    max = Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }

        public static double findMinimumNumber1( double observe[]) {

            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if ((observe[i]) < (max)) {

                    max = (observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }
        public static double findMaximumNumber1( double observe[]) {

```

```
            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max =Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }
        public static double findMaximumNumber( double observe[], double anoVal) {

            double max = anoVal;
            for (int i = 1; i < observe.length; i++) {

                if ((observe[i]) > (max)) {

                    max = (observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }



        public static double[] convertDoubleArray(String str) throws Exception {

            java.util.StringTokenizer st = new java.util.StringTokenizer(str, ",");
            String temp = "";
            java.util.ArrayList arr = new java.util.ArrayList();

            while(st.hasMoreTokens()) {

                temp = st.nextToken();
                arr.add(temp);
            }
            double d_array[] = new double[arr.size() + 1];

            for (int i = 0; i <= arr.size(); i++) {

                if (i == 0)
                    d_array[i] = 0.0;
                else {

                    try {
                        d_array[i] = convertDouble( arr.get(i - 1).toString() );
                    }
                    catch(Exception e){
                        JFrame frame = null;
                        JOptionPane.showMessageDialog(frame,
                                "Enter numerical values.",
                                "Number format error",
                                JOptionPane.ERROR_MESSAGE);
                        //  throw new FGM_HandleException();

                    }
                }
            }
            return d_array;
        }
    }
```

Annexure 4-B
Sample output



LOCATION:- Sample-Fixed depths			
PROFILE NUMBER:- 1			
ITERATION 69			
Distance (km)	**Observed anomalies (mGal)**	**Calculated anomalies (mGal)**	**Error (mGal)**
1.0	-0.043	-0.047	0.004
2.0	-0.042	-0.047	0.005
3.0	-0.04	-0.046	0.006
4.0	-0.039	-0.044	0.005
5.0	-0.036	-0.043	0.007
6.0	-0.034	-0.04	0.006
7.0	-0.03	-0.037	0.007
8.0	-0.026	-0.034	0.008
9.0	-0.021	-0.029	0.008
10.0	-0.015	-0.024	0.009
11.0	-0.008	-0.017	0.009
12.0	0.001	-0.009	0.01
13.0	0.012	0.001	0.011
14.0	0.025	0.013	0.012
15.0	0.042	0.029	0.013
16.0	0.062	0.048	0.014
17.0	0.087	0.072	0.015
18.0	0.118	0.102	0.016
19.0	0.157	0.139	0.018
20.0	0.206	0.188	0.018
21.0	0.27	0.251	0.019
22.0	0.353	0.334	0.019
23.0	0.465	0.445	0.02
24.0	0.618	0.599	0.019
25.0	0.834	0.819	0.015
26.0	1.156	1.146	0.01
27.0	1.663	1.663	0
28.0	2.534	2.549	-0.015
29.0	4.247	4.284	-0.037
30.0	9.209	9.205	0.004
31.0	16.205	16.205	0
32.0	18.958	18.96	-0.002
33.0	20.341	20.349	-0.008
34.0	21.067	21.061	0.006
35.0	21.472	21.441	0.031
36.0	21.698	21.654	0.044
37.0	21.808	21.764	0.044
38.0	21.83	21.798	0.032
39.0	21.785	21.769	0.016
40.0	21.69	21.689	0.001
41.0	21.561	21.573	-0.012
42.0	21.413	21.433	-0.02
43.0	21.257	21.28	-0.023
44.0	21.1	21.125	-0.025.

45.0	20.949	20.973	-0.024
46.0	20.807	20.829	-0.022
47.0	20.676	20.696	-0.02
48.0	20.557	20.574	-0.017
49.0	20.449	20.463	-0.014
50.0	20.351	20.363	-0.012
51.0	20.264	20.274	-0.01
52.0	20.185	20.193	-0.008
53.0	20.115	20.121	-0.006
54.0	20.052	20.057	-0.005
55.0	19.995	19.998	-0.003
56.0	19.944	19.946	-0.002
57.0	19.897	19.899	-0.002
58.0	19.855	19.856	-0.001
59.0	19.817	19.818	-0.001
60.0	19.783	19.783	0
61.0	19.752	19.751	0.001
62.0	19.723	19.721	0.002
63.0	19.697	19.695	0.002
64.0	19.672	19.67	0.002
65.0	19.65	19.648	0.002
66.0	19.63	19.627	0.003
67.0	19.611	19.608	0.003
68.0	19.594	19.591	0.003
69.0	19.578	19.574	0.004
70.0	19.563	19.559	0.004
71.0	19.549	19.545	0.004
72.0	19.536	19.532	0.004
73.0	19.524	19.52	0.004
74.0	19.513	19.509	0.004
75.0	19.502	19.498	0.004
76.0	19.492	19.488	0.004
77.0	19.483	19.479	0.004
78.0	19.474	19.47	0.004
79.0	19.466	19.462	0.004
80.0	19.459	19.454	0.005

OBJECTIVE FUNCTION = 0.01634

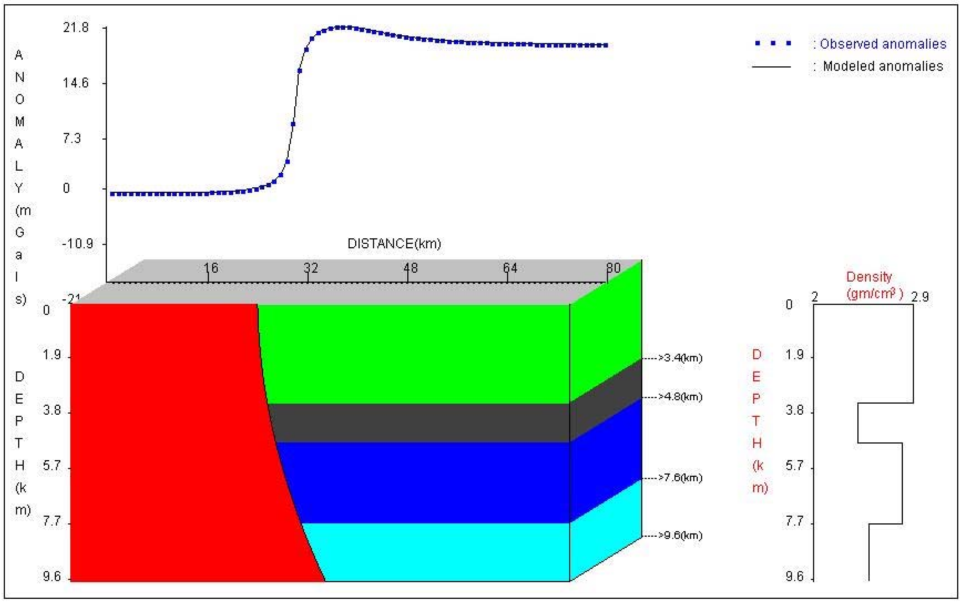
INTERPRETED PARAMETERS:

COEFFICIENTS OF THE POLYNOMIAL :-

30.03632
0.07052
0.10606

ESTIMATED DENSITIES OF THE FORMATIONS :-

2.9
2.44
2.78
2.51



LOCATION:- Sample-Fixed densities			
PROFILE NUMBER:- 1			
ITERATION 45			
Distance (km)	**Observed anomalies (mGal)**	**Calculated anomalies (mGal)**	**Error (mGal)**
1.0	-0.043	-0.041	-0.002
2.0	-0.042	-0.04	-0.002
3.0	-0.04	-0.039	-0.001
4.0	-0.039	-0.037	-0.002
5.0	-0.036	-0.035	-0.001
6.0	-0.034	-0.032	-0.002
7.0	-0.03	-0.029	-0.001
8.0	-0.026	-0.025	-0.001
9.0	-0.021	-0.021	-0
10.0	-0.015	-0.015	-0
11.0	-0.008	-0.008	-0
12.0	0.001	0.001	0
13.0	0.012	0.011	0.001
14.0	0.025	0.024	0.001
15.0	0.042	0.04	0.002
16.0	0.062	0.06	0.002
17.0	0.087	0.084	0.003
18.0	0.118	0.114	0.004
19.0	0.157	0.153	0.004
20.0	0.206	0.202	0.004
21.0	0.27	0.265	0.005
22.0	0.353	0.348	0.005
23.0	0.465	0.459	0.006
24.0	0.618	0.612	0.006
25.0	0.834	0.83	0.004
26.0	1.156	1.153	0.003
27.0	1.663	1.663	0
28.0	2.534	2.537	-0.003
29.0	4.247	4.254	-0.007
30.0	9.209	9.208	0.001
31.0	16.205	16.208	-0.003
32.0	18.958	18.959	-0.001
33.0	20.341	20.344	-0.003
34.0	21.067	21.059	0.008
35.0	21.472	21.453	0.019
36.0	21.698	21.682	0.016
37.0	21.808	21.803	0.005
38.0	21.83	21.838	-0.008
39.0	21.785	21.802	-0.017
40.0	21.69	21.711	-0.021.

41.0	21.561	21.582	-0.021
42.0	21.413	21.429	-0.016
43.0	21.257	21.266	-0.009
44.0	21.1	21.103	-0.003
45.0	20.949	20.946	0.003
46.0	20.807	20.8	0.007
47.0	20.676	20.666	0.01
48.0	20.557	20.545	0.012
49.0	20.449	20.435	0.014
50.0	20.351	20.338	0.013
51.0	20.264	20.25	0.014
52.0	20.185	20.173	0.012
53.0	20.115	20.103	0.012
54.0	20.052	20.041	0.011
55.0	19.995	19.985	0.01
56.0	19.944	19.935	0.009
57.0	19.897	19.89	0.007
58.0	19.855	19.849	0.006
59.0	19.817	19.813	0.004
60.0	19.783	19.779	0.004
61.0	19.752	19.749	0.003
62.0	19.723	19.721	0.002
63.0	19.697	19.696	0.001
64.0	19.672	19.673	-0.001
65.0	19.65	19.651	-0.001
66.0	19.63	19.632	-0.002
67.0	19.611	19.614	-0.003
68.0	19.594	19.597	-0.003
69.0	19.578	19.582	-0.004
70.0	19.563	19.567	-0.004
71.0	19.549	19.554	-0.005
72.0	19.536	19.542	-0.006
73.0	19.524	19.53	-0.006
74.0	19.513	19.52	-0.007
75.0	19.502	19.51	-0.008
76.0	19.492	19.5	-0.008
77.0	19.483	19.491	-0.008
78.0	19.474	19.483	-0.009
79.0	19.466	19.476	-0.01
80.0	19.459	19.468	-0.009

OBJECTIVE FUNCTION = 0.00485

INTERPRETED PARAMETERS:

COEFFICIENTS OF THE POLYNOMIAL :-

30.03018
0.07627
0.11048

ESTIMATED DEPTHS TO THE INTERFACES :-

3.44
4.8
7.59
9.57

```
package com.mod2dgrexp.view;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;

import javax.swing.JFrame;
import javax.swing.JOptionPane;


import com.mod2dgrexp.control.MOD2DGREXP_Controller;
import com.mod2dgrexp.model.MOD2DGREXP_CalculateValues;

public class MOD2DGREXP_MainView extends Frame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static void main(String s[]) {

        MOD2DGREXP_MainView cm = new MOD2DGREXP_MainView();
        cm.setSize(1280, 768);
        cm.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                JFrame frame = null;

                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit MOD2DGREXP ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(MOD2DGREXP_Controller.success==false){
                        String fileName = MOD2DGREXP_CalculateValues.input_area_name+".jpg"
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        });
        cm.setTitle("MOD2DGREXP");
        cm.setResizable(false);
        cm.add(new MOD2DGREXP_MainPanel());
        cm.setVisible(true);

    }

}
------------------------------------------------------------------------------------------

package com.mod2dgrexp.view;

import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import javax.swing.JFileChooser;

import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;



public class MOD2DGREXP_MainPanel extends Panel {

    /**
     *
```

```
     */
    private static final long serialVersionUID = 1L;

    Panel p_North, p_West,p_South;
    public static TextArea img = new TextArea(36,135);
    public static Panel p_East,p_Center;
    static TextField inputValues [] = new TextField[12];
    static TextArea graphValues = new TextArea(38,30);
    Button actionButton[] = new Button[5];
    Object rowdata[][]={};

    /**Field Area Name*/
    final static int AREA_FE = 0;
    /**Profile Name*/
    final static int PROFILE_NAME = 1;
    /**Number of observation*/
    public static final int N_OBS = 2 ;
    /**Distance(km)*/
    public static final int X_KM = 3;
    /**Number of Observed values*/
    public static final int NOB_GOB = 4;
    /** SD */
    public static final int SD_POLY = 5;
    /**LAMBDA*/
    public static final int LAMBDA_VAL = 6 ;
    /**ZMIN(km)*/
    public static final int DEP_ZMIN = 7;
    /**ZMAX(km)*/
    public static final int DEP_ZMAX = 8;
    /**Number of iteration values*/
    public static final int NOB_ITER = 9;


    public MOD2DGREXP_MainPanel() {

        this.setLayout(new BorderLayout());
        p_North = new Panel();
        p_West = new Panel();
        p_East = new Panel ();
        p_South = new Panel();
        p_Center = new Panel();

        Label graphLabel = new Label("AUTOMATIC MODELING OF GRAVITY ANOMALIES OF 2D SEDIMENTARY BASINS ", Label.CENTER);
        Label graphLabel1 = new Label(" USING EXPONENTIAL DENSITY MODEL", Label.CENTER);
        graphLabel.setFont(new Font("Arial", 10, 18));
        graphLabel1.setFont(new Font("Arial", 10, 18));
        p_Center.add(graphLabel);
        p_Center.add(graphLabel1);

        for(int i = 0; i < 11; i++){
            inputValues[i] = new TextField();
        }
        p_North.setFont(new Font("Bold",1,12));
        actionButton[0] = new Button("Load data");
        actionButton[1] = new Button("Modeling");
        actionButton[2] = new Button("Save and Print");
        actionButton[3] = new Button("Clear");
        actionButton[4] = new Button("Exit");

        this.populateNorthPanel();
        MOD2DGREXP_TableView.populateEastPanel(rowdata);
        this.add(p_North, BorderLayout.NORTH);
        this.add(p_West, BorderLayout.WEST);
        this.add(p_East, BorderLayout.EAST);
        this.add(p_South, BorderLayout.SOUTH);
        p_Center.setSize(1000, 760);
        this.add(p_Center, BorderLayout.CENTER);
        img.setEditable(false);
        p_Center.add(img);
        this.setVisible(true);
    }

    public void populateNorthPanel(){
        p_North.setLayout(new GridLayout(5,6));
```

```
        p_North.add(new Label("Area Name"));
        p_North.add(inputValues[0]);
        p_North.add(new Label("Profile Name"));
        p_North.add(inputValues[1]);
        p_North.add(new Label("Number of observation:"));
        p_North.add(inputValues[2]);
        p_North.add(new Label("Distance (km)"));
        p_North.add(inputValues[3]);
        //p_North.add(new Label("Elevation of each station (km)"));
        //p_North.add(inputValues[4]);
        p_North.add(new Label("Observed anomalies (mGal)"));
        p_North.add(inputValues[4]);
        p_North.add(new Label("Surface density contrast (gm/cc)"));
        p_North.add(inputValues[5]);
        p_North.add(new Label("Lambda (1/km)"));
        p_North.add(inputValues[6]);
        p_North.add(new Label("Minimum depth(km):"));
        p_North.add(inputValues[7]);
        p_North.add(new Label("Maximum depth(km):"));
        p_North.add(inputValues[8]);
        p_North.add(new Label("Iterations"));
        p_North.add(inputValues[9]);
        p_North.add(new Label(""));
        p_North.add(new Label(""));
        p_North.add(new Label(""));
        p_North.add(new Label(""));


        p_North.add(actionButton[0]);
        p_North.add(actionButton[1]);
        p_North.add(actionButton[2]);
        p_North.add(actionButton[3]);
        p_North.add(actionButton[4]);

        actionButton[0].addActionListener(new
    com.mod2dgrexp.control.MOD2DGREXP_Controller());
        actionButton[1].addActionListener(new
    com.mod2dgrexp.control.MOD2DGREXP_Controller());
        actionButton[2].addActionListener(new
    com.mod2dgrexp.control.MOD2DGREXP_Controller());
        actionButton[3].addActionListener(new
    com.mod2dgrexp.control.MOD2DGREXP_Controller());
        actionButton[4].addActionListener(new
    com.mod2dgrexp.control.MOD2DGREXP_Controller());

    }

    public static HashMap captureValues(){

        HashMap h_Map = new HashMap();

        try {

            h_Map.put("N_OBS", inputValues[N_OBS].getText());
            h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
            h_Map.put("LAMBDA_VAL",inputValues[LAMBDA_VAL].getText());
            h_Map.put("DEP_ZMIN", inputValues[DEP_ZMIN].getText());
            h_Map.put("DEP_ZMAX", inputValues[DEP_ZMAX].getText());
            h_Map.put("X_KM", inputValues[X_KM].getText());
            h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
            h_Map.put("NOB_ITER", inputValues[NOB_ITER].getText());
            h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
            h_Map.put("PROFILE_NAME",inputValues[PROFILE_NAME].getText());
        }
        catch (Exception e) {
            e.printStackTrace();
        }

        return h_Map;

    }

    public static void clearPanel(TextArea p) {

```

```java
            Graphics g = p.getGraphics();
            g.setColor(Color.WHITE);
            g.fillRect(0, 0, 1280, 650);
        }



        public static void loadData1()throws IOException  {
            try{

                String current = System.getProperty("user.dir");
                JFileChooser chooser=new  JFileChooser(current);
                int returnVal = chooser.showOpenDialog(null);
                String dis[],gobs[];
                String disval = "" ,gobsval="";
                Workbook w;

                if(returnVal == JFileChooser.APPROVE_OPTION) {
                    File f = chooser.getSelectedFile();
                    w = Workbook.getWorkbook(f);
                    Sheet sheet = w.getSheet(0);
                    dis = new String[sheet.getRows()+1];
                    //ele = new String[sheet.getRows()+1];
                    gobs = new String[sheet.getRows()+1];
                    for (int j = 0; j < sheet.getColumns(); j++) {
                        for (int i = 1; i < sheet.getRows(); i++) {
                            Cell cell = sheet.getCell(j, i);
                            CellType type = cell.getType();
                            if (type == CellType.LABEL) {
                                //  System.out.println("I got a label "
                                //    + cell.getContents());
                                MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.AREA_FE]
etText(cell.getContents());

                            }

                            if (type == CellType.NUMBER) {
                                if (j == 1){

                                    MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.PROF
E_NAME].setText(cell.getContents());

                                }
                                if (j == 2){

                                    MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.N_OB
.setText(cell.getContents());

                                }

                                if (j == 3){

                                    dis[i] = cell.getContents()+",";
                                    disval = disval + dis[i];
                                }
                                //if (j == 4){

                                //   ele[i] = cell.getContents()+",";
                                //  eleval = eleval + ele[i];
                                //}
                                if (j == 4){

                                    gobs[i] = cell.getContents()+",";
                                    gobsval = gobsval+gobs[i];
                                }

                                if (j == 5){

                                    MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.SD_P
Y].setText(cell.getContents());

                                }
                                if (j == 6){

```

```
                              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.LAMBDA_VAL].setText(cell.getContents());

                          }

                          if (j == 7){

                              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.DEP_ZMIN].setText(cell.getContents());

                          }
                          if (j == 8){


                              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.DEP_ZMAX].setText(cell.getContents());
                          }
                          if (j == 9){
                              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.NOB_ITER].setText(cell.getContents());

                          }

                          //System.out.println("I got a number "
                          //  + cell.getContents());
                      }

                  }
              }
              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.X_KM].setText(""+disvl);
              MOD2DGREXP_MainPanel.inputValues[MOD2DGREXP_MainPanel.NOB_GOB].setText(""+gbsval);
          }
      }
      catch (BiffException e) {
          e.printStackTrace();
      }



  }


  public static void clearDefaultValues() {

      inputValues[N_OBS].setText("");
      inputValues[SD_POLY].setText("");
      inputValues[LAMBDA_VAL].setText("");
      inputValues[DEP_ZMIN].setText("");
      inputValues[DEP_ZMAX].setText("");
      inputValues[X_KM].setText("");
      inputValues[NOB_GOB].setText("");
      inputValues[NOB_ITER].setText("");
      inputValues[AREA_FE].setText("");
      inputValues[PROFILE_NAME].setText("");
  }

}
-----------------------------------------------------------------------------------------

package com.mod2dgrexp.view;

import java.applet.Applet;
import java.awt.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.text.DecimalFormat;
import com.mod2dgrexp.model.MOD2DGREXP_CalculateValues;
import com.mod2dgrexp.util.MOD2DGREXP_Utility;


public class MOD2DGREXP_DrawGraph extends Applet {


```

```
        /**
         *
         */
        private static final long serialVersionUID = 1L;
        float maxZ, maxX,maxY;
        double inidep;
        public void drawGraph(Graphics2D g){
            g.setColor(Color.black);
            g.drawLine(150,50,150,550);
            g.drawLine(90,45 ,1040,45);
            g.drawString("DISTANCE(Km)",335 ,295);
            String a[] = {" A ", " N " , " O " , " M ", " A ", " L ", " Y ", "(m ", " G "," a "," l ", " s)"};
            String b[] = {" D ", " E ", " P ", " T ", " H ", "(k ", " m)"};
            for (int i = 0;i < a.length; i++) {

                g.drawString(" "+ a[i], 100, 60 + (i * 20));
            }
            for (int i = 0;i < b.length; i++) {

                g.drawString(" "+b[i], 100, 350 + (i * 20));
            }

        }

        public void plot(Graphics2D g){
            DecimalFormat df = new DecimalFormat("0.#");
            DecimalFormat df1 = new DecimalFormat("0.##");
            maxX = (float) MOD2DGREXP_Utility.findMaximumNumber1(MOD2DGREXP_CalculateValues.input_x_km);
            maxY = MOD2DGREXP_Utility.findMaximumNumber(MOD2DGREXP_CalculateValues.input_nob_gob);
            inidep  = MOD2DGREXP_Utility.findMaximumNumber1(MOD2DGREXP_CalculateValues.z);
            maxZ = (float) inidep;
            g.setColor(Color.BLACK);
            g.drawString("0",140,60);
            g.drawString(""+(int)maxY  ,125,300);

            g.drawString("0", 125,310);
            g.drawString("|", 600, 308);
            g.drawString(""+df.format(MOD2DGREXP_CalculateValues.input_x_km[MOD2DGREXP_CalculateValues.input_n_obs]) ,600,320);
            g.draw(new Line2D.Float(150, 300, 600, 300));
            float points = maxX / 5;

            int yInterval = 50;
            int zInterval = 50;

            float xInterval=(float)(MOD2DGREXP_CalculateValues.input_x_km[MOD2DGREXP_CalculateValues.input_n_obs]/5);
            float xplot=0;

            for (float x = xInterval, j =1; x < 600; x+=xInterval){

                xplot = xplot + xInterval;
                if(j > 4)
                    break;
                g.drawString("|",(float) (150 + (450 * x / maxX)), 308);
                g.drawString("" + df.format(xplot), (float) (150 + (450 * x / maxX)) - 3, 323);
                j++;
            }

            points = maxY / 5;
            for (int x = yInterval, j = 1; x < 250; x += yInterval){
                g.drawString("-", 148, 50 + x);
                g.drawString("" + (int)(points * j), 125, 50 + x);
                j++;
            }
            float point = maxZ / 5 ;

            for(int x = zInterval+250,j =1; x < 550; x+=zInterval){

                if(j > 4)
                    break;
                g.drawString("-", 148, 50 + x);
```

```
                g.drawString("" +df.format(point * j), 125, 50 + x);
                j++;
            }
            g.drawString("-", 148, 552);
            g.drawString(""+df1.format(maxZ), 125, 550);
    }


    public void plotXYCoordinates (Graphics2D g){

            float prevx = 150;
            float prevy = 50;
            float xpoint = 0;
            float ypoint = 0;
            float gypoint = 0;
            for (int k = 1; k <= MOD2DGREXP_CalculateValues.input_n_obs; k++){
                xpoint = (float)(450 * MOD2DGREXP_CalculateValues.input_x_km[k]/ maxX) ;
                ypoint = (float)(250 * MOD2DGREXP_CalculateValues.gc[k]/ maxY);
                gypoint = (float)(250 * MOD2DGREXP_CalculateValues.input_nob_gob[k]/ maxY);
                g.setColor(Color.BLACK);
                g.draw(new Line2D.Float(prevx, prevy, 150 + xpoint, 50 + ypoint));
                g.setColor(Color.BLUE);
                g.setFont(new Font("Arial", 20, 55));
                g.drawString(".",150 + xpoint - 6 , 50 + gypoint);
                prevx = 150 + xpoint;
                prevy = 50 + ypoint;

            }
    }

    public void plotZCoordinates (Graphics2D g){
            float prevx = 150;
            float prevz = 300 + (float)(250 * MOD2DGREXP_CalculateValues.input_zmin_km / maxZ );;
            float xpoint = 0;
            float zpoint = 0;
            GradientPaint gradient = new GradientPaint(10, 10, Color.yellow, 30, 200, Color.MAGENTA, true);
            g.setPaint(gradient);
            g.fill(new Rectangle2D.Float(151 , 300, 450, 250 ));
            for (int k = 1; k <= MOD2DGREXP_CalculateValues.input_n_obs; k++) {

                xpoint = (float)(450 * MOD2DGREXP_CalculateValues.input_x_km[k] /  maxX);
                zpoint = (float)(250 * MOD2DGREXP_CalculateValues.z[k] / maxZ );
                float vary=prevx;
                g.setColor(Color.red);
                if(prevz <= 300 + zpoint){
                    while(vary<= 150 + xpoint){
                        g.draw(new Line2D.Float(prevx, prevz, vary,300 + zpoint));
                        vary = (float) (vary + 0.001);
                    }
                    g.fill(new Rectangle2D.Float(prevx , 300 + zpoint,(150 + xpoint) - prevx, 250 - zpoint ));
                }
                else{
                    vary = prevz;
                    while( 300 + zpoint <= vary){
                        g.draw(new Line2D.Float(prevx, prevz,150 + xpoint, vary));
                        vary = (float) (vary - 0.001);
                    }
                    g.fill(new Rectangle2D.Float(prevx ,prevz,(150 + xpoint)- prevx, 550 - prev
));
                }
                g.setColor(Color.black);
                g.draw(new Line2D.Float(prevx, prevz,150 + xpoint,300 + zpoint));
                prevx = 150 + xpoint;
                prevz = 300 + zpoint;

            }
            g.setColor(Color.white);
            g.fill(new Rectangle2D.Float(151 ,550,450, 50 ));


    }

```

```java
    public void drawOBJ(Graphics2D g2) {

        g2.setColor(Color.BLACK);
        g2.drawLine(780, 45, 780, 560);
        g2.drawLine(90, 560, 1040, 560);
        g2.drawLine(1040, 45, 1040, 560);
        g2.drawLine(90, 45, 90, 560);

        g2.drawLine(820, 70, 820, 160);
        g2.drawLine(820, 160, 910, 160);
        g2.drawString("J", 800, 90);

        double maxOb = MOD2DGREXP_Utility.findMaximumNumber1(MOD2DGREXP_CalculateValues.o_funct) ;
        int ini = MOD2DGREXP_Utility.findMaximumNumber(MOD2DGREXP_CalculateValues.o_iter);
        if(ini == 5)
            ini= ini+1;
        int maxiter = ( ini / 3 * 5 ) * 2;
        int point;
        int xInterval = 22;
        point = ( ( ini ) / 3 * 5 ) / 5;

        for (int x = xInterval, j = 1; x < 90 ; x += xInterval) {

            g2.drawString("'", 821 + x, 170);
            g2.drawString("" + (point * j), 820 + x - 3, 175);
            j++;
        }
        float prevx = 820;
        float prevy = 70;
        float xpoint = 0;
        float ypoint = 0;

        for (int i = 1; i <= MOD2DGREXP_CalculateValues.o_iter; i++) {
            xpoint = (float)( 250 * i /maxiter );
            ypoint = 70 - (float) ( ( 90 * (MOD2DGREXP_CalculateValues.o_funct[i]) / maxOb );
            if(i==MOD2DGREXP_CalculateValues.o_iter){
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint - 4, 90 + ypoint));
            }
            else {
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 90 + ypoint));
            }
            prevx = 820 + xpoint;
            prevy = 90 + ypoint;

        }
        DecimalFormat d1= new DecimalFormat("0.###");
        DecimalFormat d= new DecimalFormat("0.#");
        g2.drawString(""+d.format(MOD2DGREXP_CalculateValues.o_funct[1]), 780, 70);
        g2.drawString("
"+d1.format(MOD2DGREXP_CalculateValues.o_funct[MOD2DGREXP_CalculateValues.o_iter]), 820 + xpoint, 90 + ypoint);
        g2.setFont(new Font("Arial", 40,11));
        g2.drawString ("Iterations",850,186);
    }


    public void drawSd(Graphics2D g2) {

        g2.setColor(Color.black);
        g2.drawLine(780, 200, 1040, 200);
        g2.setColor(Color.red);
        g2.setFont(new Font("Arial", 20, 12));
        DecimalFormat d= new DecimalFormat("0.##");
        DecimalFormat d1= new DecimalFormat("0.###");
        g2.draw(new Line2D.Float(820, 300, 820, 550));
        g2.drawString(""+d.format(inidep),790,550);
        g2.drawString("-",820,550+2);
        g2.drawString("0",807 ,300);
        g2.drawLine(820, 300, 910, 300);
        double maxOb1 = MOD2DGREXP_Utility.findMaximumNumber1(MOD2DGREXP_CalculateValues.vsd);
        DecimalFormat df = new DecimalFormat("0.#");
        float points = (float)inidep / 5 ;
```

```
            int zInterval = 50;
            for(int x = zInterval+250,j = 1; x < 550; x+=zInterval){

                if(j > 4)
                    break;

                g2.drawString("-",820,50 + x + 2);
                g2.drawString("" +df.format(points * j), 790, 50 + x);
                j++;

            }

            float prevx = 820 + (float) ( ( 90 * ( Math.abs(MOD2DGREXP_CalculateValues.vsd[1] ) / maxOb1 ) );
            float prevy = 300 + (float)(250 * MOD2DGREXP_CalculateValues.input_zmin_km / maxZ )
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= MOD2DGREXP_CalculateValues.count; i++) {

                xpoint = (float)( 90 * Math.abs(MOD2DGREXP_CalculateValues.vsd[i]) / maxOb1 );
                ypoint = (float)( 250 * MOD2DGREXP_CalculateValues.dep[i] / inidep );
                g2.setColor(Color.blue);
                g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 300 + ypoint));
                prevx = 820 + xpoint;
                prevy = 300 + ypoint;
            }
            g2.drawString(""+d.format(MOD2DGREXP_CalculateValues.vsd[1] ),805+ (float) ( ( 90 * ( Math.abs(MOD2DGREXP_CalculateValues.vsd[1] )) / maxOb1 ) ) ,300 );
            g2.drawString(""+d1.format(MOD2DGREXP_CalculateValues.vsd[MOD2DGREXP_CalculateValues.count] ),820+ (float) ( ( 90 * ( Math.abs(MOD2DGREXP_CalculateValues.vsd[MOD2DGREXP_CalculateValues.count] )) / maxOb1 ) ), 550);
            g2.setColor(Color.BLACK);
            g2.drawString("Variation of density contrast " , 800,220);
            g2.drawString("with depth" , 850,240);
            g2.setFont(new Font("Arial", 40,11));
            g2.drawString ("Density contrast",830,285);
            g2.drawString ("(gm/cc)",843,295);
            g2.drawString("Z(km)", 790,(float)( 300+250/2));
        }

        public void idex(Graphics2D g){
            g.setColor(Color.BLUE);
            g.setFont(new Font("Arial", 20, 50));
            g.drawString(" ...     ",595,70);
            g.setFont(new Font("Arial", 20, 12));
            g.drawString("Observed anomalies",650,70);
            g.setColor(Color.BLACK);
            g.drawString("____:",615,87);
            g.drawString("Calculated anomalies",650,90);
            g.setColor(Color.black);
            g.drawString("<----------------------------------------------------:",450,340);
            g.setColor(Color.RED);
            g.drawString("Estimated Depth ",660,340);
            g.drawString(" Structure",660,355);
        }

    }

    ---------------------------------------------------------------------------------------------

    package com.mod2dgrexp.view;

    import java.awt.Dimension;

    import javax.swing.JScrollPane;
    import javax.swing.JTable;

    public class MOD2DGREXP_TableView {

        public static  void populateEastPanel(Object rowData[][]) {
            com.mod2dgrexp.view.MOD2DGREXP_MainPanel.p_East.removeAll();
            Object columnNames[] = {"Distance(km)", "Observed anamolies (mGal)", "Calculated
```

```java
anamolies (mGal)","Depth(km)"};
            JTable table = new JTable(rowData, columnNames);
            table.setPreferredScrollableViewportSize(new Dimension(300,550));
            JScrollPane scrollPane = new JScrollPane(table);
            scrollPane.setAutoscrolls(true);
            com.mod2dgrexp.view.MOD2DGREXP_MainPanel.p_East.add(scrollPane);
            com.mod2dgrexp.view.MOD2DGREXP_MainPanel.p_East.validate();
            com.mod2dgrexp.view.MOD2DGREXP_MainPanel.p_East.setVisible(true);
        }

}
---------------------------------------------------------------------------------------------

package com.mod2dgrexp.model;

import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.text.DecimalFormat;
import java.util.HashMap;
import javax.imageio.ImageIO;
import com.mod2dgrexp.util.MOD2DGREXP_Utility;
import com.mod2dgrexp.view.MOD2DGREXP_DrawGraph;
import com.mod2dgrexp.view.MOD2DGREXP_MainPanel;


public class MOD2DGREXP_CalculateValues {

    public static Object obj[][] = null;
    public static double []o_funct;
    public static int o_iter,count  ;
    public static double input_zmax_km = 0;
    public static double input_zmin_km = 0;
    public static int input_n_obs=0 ;
    public static double []input_x_km;
    public static double []input_nob_gob;
    public static double []gc;
    public static double []z;
    public static double []vsd = null;
    public static double []dep = null;
    public static String input_area_name = "";
    public static String input_profile="";
    static BufferedImage image;

    public void getAnamolyValues(HashMap h_Map) {

        double PI = 22.0/7.0;
        double GK = 13.3333;
        double input_sd_poly = 0;
        double input_lambda_val = 0;
        int input_nob_iter = 0;
        try {

            input_n_obs =  MOD2DGREXP_Utility.convertInteger((String)h_Map.get("N_OBS"));
            input_sd_poly = MOD2DGREXP_Utility.convertDouble((String)h_Map.get("SD_POLY"));
            input_lambda_val =
MOD2DGREXP_Utility.convertDouble((String)h_Map.get("LAMBDA_VAL"));
            input_zmin_km = MOD2DGREXP_Utility.convertDouble((String)h_Map.get("DEP_ZMIN"))
            input_zmax_km = MOD2DGREXP_Utility.convertDouble((String)h_Map.get("DEP_ZMAX"))
            input_x_km = MOD2DGREXP_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
            input_nob_gob =
MOD2DGREXP_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
            input_nob_iter =
MOD2DGREXP_Utility.convertInteger((String)h_Map.get("NOB_ITER"));
            input_area_name =
MOD2DGREXP_Utility.convertString((String)h_Map.get("AREA_FE"));
            input_profile =
MOD2DGREXP_Utility.convertString((String)h_Map.get("PROFILE_NAME"));
        }
        catch (Exception e) {
            e.printStackTrace();
        }

```

```
        double ALER = 0.001*input_n_obs;
        o_funct = new double[input_nob_iter+1];
        gc = new double[input_n_obs+1];
        double err[] = new double[input_n_obs+1];
        z = new double[input_n_obs+2];
        double dcz[] = new double [input_n_obs+1];

        z[1]=0.0001;
        z[input_n_obs]=0.0001;
        for (int kk = 2; kk <= input_n_obs - 1; kk++) {
            if(input_lambda_val==0)
                z[kk]=input_nob_gob[kk]/(GK*PI*input_sd_poly);
            else
                z[kk]=-(1 / input_lambda_val) * Math.log(1 - ((input_lambda_val * input_nob_gob[kk]) / (GK * PI * input_sd_poly)));

        }

        GF(input_n_obs,input_x_km,z,input_sd_poly,input_lambda_val,GK,gc);

        double funct1 = 0;

        for (int k = 1;k <= input_n_obs; k++) {

            err[k] = input_nob_gob[k] - gc[k];
            funct1 = funct1 + Math.pow(input_nob_gob[k] - gc[k], 2);
        }
        funct1=Math.sqrt(funct1 /input_n_obs);

        for (int ITER = 1;ITER <= input_nob_iter; ITER++) {
            for(int kk = 2; kk <= input_n_obs - 1; kk++) {
                err[kk] = input_nob_gob[kk] - gc[kk];
                if(input_lambda_val == 0)
                    z[kk]= z[kk]+err[kk]/(GK*PI*input_sd_poly);
                else{
                    double st = GK*PI*input_sd_poly*Math.exp(-input_lambda_val*z[kk]);
                    dcz[kk]=-(1/input_lambda_val)*Math.log(1-((input_lambda_val*err[kk])/st);
                    z[kk]=z[kk]+0.5*dcz[kk];
                }
                if (z[kk] <= input_zmin_km)
                    z[kk] = input_zmin_km;

                if (z[kk] > input_zmax_km)
                    z[kk] = input_zmax_km;

            }
            GF(input_n_obs,input_x_km,z,input_sd_poly,input_lambda_val,GK,gc);

            double funct2 = 0;
            for (int LI = 1; LI <= input_n_obs; LI++) {

                funct2 = funct2 + Math.pow((input_nob_gob[LI] - gc[LI]), 2);

            }
            funct2 = Math.sqrt(funct2/input_n_obs);
            o_iter = ITER;
            o_funct[ITER] = funct1;
            setGraphValues(input_n_obs,o_iter, input_x_km,z, input_nob_gob, gc,  funct1, input_area_name);
            denCal(input_sd_poly,input_lambda_val);
            drawGraph();

            if (funct2 - funct1 < 0 || funct2 - funct1 == 0) {

                if (funct2 - ALER <= 0) {

                    break;
                }

                else if (funct2 - ALER > 0) {

                    funct1 = funct2;
                }
```

```
            }
            else if (funct2 - funct1 > 0) {

                break;
            }
        }

    }

    public static void denCal(double sd,double lambda){

        int i = 1;
        double z1=MOD2DGREXP_CalculateValues.input_zmin_km;
        double z2 = MOD2DGREXP_Utility .findMaximumNumber1(MOD2DGREXP_CalculateValues.z);
        vsd = new double[(int) Math.pow(input_n_obs,2)];
        dep = new double[(int) Math.pow(input_n_obs,2)];
        while(z1 <= z2){

            double dc =  sd*Math.exp(-lambda*z1);
            vsd[i] = dc;
            dep[i] = z1;

            z1 = z1+0.1;
            i++;
        }

        count = i;
        vsd[count] = sd*Math.exp(-lambda*z2);
        dep[count] = z2;

    }


    public double [] GF(int n,double []x,double []zv,double sd,double la,double gk,double
[]gc) {

        double ggc=0;
        double []xx = new double [n+2];
        double []zt = new double[10000];
        double []x1 = new double[10000];
        double []gs = new double[10000];

        for(int JJ = 1 ; JJ <= n ;JJ++){
            gc[JJ]=0.0;
        }
        for(int k1=1;k1<=n;k1++){
            for(int k2=1;k2<=n;k2++){
                xx[k2]=x[k2]-x[k1];
            }
            xx[n+1]=xx[1];
            zv[n+1]=zv[1];
            double grav = 0;
            for(int i=1;i<=n;i++){
                double dxx = xx[i+1] - xx[i];
                double dzz = zv[i+1] - zv[i];
                double r = Math.sqrt(Math.pow(dxx,2)+Math.pow(dzz,2));
                double c = dxx / r;
                double s = dzz / r;
                double ct = c / s;
                double dx = (x[2]-x[1])/4;
                double ZB = Math.abs(zv[i+1]-zv[i]);
                int nd = (int)(ZB/dx)+1;
                double n1 = nd / 2;
                if (nd - (2 * n1 ) < 0 || nd - ( 2 * n1 ) > 0) {
                    nd = nd + 1;
                }
                double DZ = ZB / nd;
                int N2 = nd + 1;

                if(zv[i+1]-zv[i]==0)
                    break;
                for(int JZ=1;JZ<=N2;JZ++){
                    if(zv[i+1]-zv[i]<0) {
                        zt[JZ] = zv[i] - DZ * (JZ-1);
```

```
                    }
                    if(zv[i+1]-zv[i]>0)     {
                        zt[JZ] = zv[i] + DZ * (JZ-1);
                    }

                    if(zt[JZ]<0)
                        zt[JZ] = 0;
                    x1[JZ] = xx[i]+(zt[JZ]-zv[i]) * ct;
                    if(Math.abs(x1[JZ])<0.01)
                        x1[JZ] = 0;

                }

                for(int JZ=1;JZ<=N2;JZ++){
                    double DC = (sd * Math.exp(-la * zt[JZ]));
                    double a = xx[i] - zv[i] * ct;
                    double anum = a + zt[JZ] * ct;
                    gs[JZ]= -13.3333*DC*Math.atan(anum / zt[JZ]);

                }
                ggc=SIMP(gs,zt,N2,ggc);
                grav = grav + ggc;

            }

            gc[k1] = grav;


        }
        return gc;
    }

    public double SIMP(double []gs,double []z,int n,double ggc) {

        double dz = z[2]-z[1];
        double sum1 = 0.0;
        double sum2 = 0.0;
        int n1 = n / 2;
        int n4 = n1 - 1;
        for(int I = 1; I <= n1; I++) {
            int n2 = 2 * I;
            sum1 = sum1 + gs[n2];
        }
        for(int I = 1; I <= n4; I++) {
            int n3 = 2 * I +1;
            sum2 = sum2 + gs[n3];
        }
        ggc = gs[1]+4*sum1+2*sum2+gs[n];
        ggc = ggc * dz / 3.0;
        return ggc;
    }

    public static void  setGraphValues(int i_no_obs,int ite,double []dis,double []dep,doub
[]GOBS,double []GCAL,double FUNCT,String Area_fe) {

        obj = new Object[i_no_obs+21][4];
        DecimalFormat df =new DecimalFormat("0.###");
        DecimalFormat d1 =new DecimalFormat("0.#######");
        for(int K=1;K<=i_no_obs;K++){
            obj[K][0]= "" + dis[K];
            obj[K][1]= "" + df.format(GOBS[K]);
            obj[K][2]= "" + df.format(GCAL[K]);
            obj[K][3]= "" + df.format(dep[K]);
        }

        obj[0][0] ="ITERATION";
        obj[0][1] = "=" +" "+ite;
        obj[i_no_obs+2][0] = "OBJECTIVE " ;
        obj[i_no_obs+2][1] = "FUNCTION =";
        obj[i_no_obs+2][2] = d1.format(FUNCT);
    }

    public static void drawGraph(){
        final MOD2DGREXP_DrawGraph dg = new MOD2DGREXP_DrawGraph();
        try
```

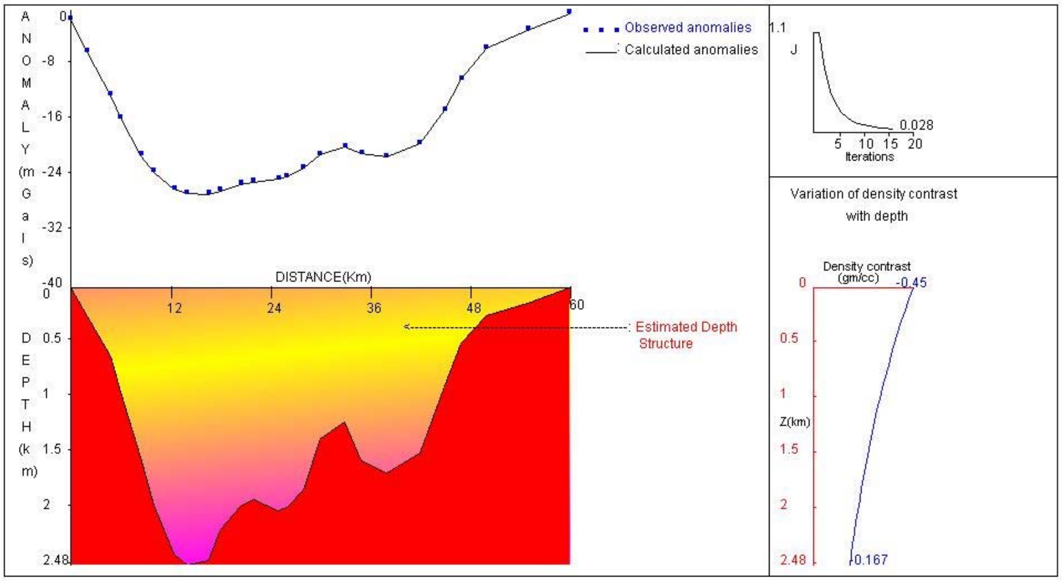
```java
        {
            int width = 1280;
            int height = 650;
            BufferedImage buffer = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
            Graphics g1= buffer.createGraphics();
            g1.setColor(Color.WHITE);
            g1.fillRect(0,0,width,height);
            Graphics2D g2 = (Graphics2D)g1 ;
            dg.plot(g2);
            dg.plotXYCoordinates(g2);
            dg.plotZCoordinates(g2);
            dg.idex(g2);
            dg.drawGraph(g2);
            dg.plot(g2);
            dg.drawOBJ(g2);
            dg.drawSd(g2);
            FileOutputStream os = new FileOutputStream(MOD2DGREXP_CalculateValues.input_area_name+".jpg");
            ImageIO.write(buffer, "jpg", os);
            os.close();

            String path = MOD2DGREXP_CalculateValues.input_area_name+".jpg";
            image = ImageIO.read(new File(path));

            Graphics g_image = MOD2DGREXP_MainPanel.img.getGraphics();
            g_image.drawImage(image, -80, -40,image.getWidth(), image.getHeight(),dg);

            MouseListener ml3 = new MouseAdapter(){
                public void mouseClicked(MouseEvent e){
                    Graphics g_image = MOD2DGREXP_MainPanel.img.getGraphics();
                    g_image.drawImage(image, -80,-40,image.getWidth(), image.getHeight(),dg);
                }
            };
            MOD2DGREXP_MainPanel.img.addMouseListener(ml3);
        }
        catch (Exception e2) {

            e2.printStackTrace();
        }
    }
}
-------------------------------------------------------------------------------------------

package com.mod2dgrexp.control;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import com.mod2dgrexp.model.MOD2DGREXP_CalculateValues;
import com.mod2dgrexp.view.MOD2DGREXP_DrawGraph;
import com.mod2dgrexp.view.MOD2DGREXP_MainPanel;
import com.mod2dgrexp.view.MOD2DGREXP_TableView;


public class MOD2DGREXP_Controller implements ActionListener {

    MOD2DGREXP_DrawGraph dg = new MOD2DGREXP_DrawGraph();
    Object rowdata[][]={};
    public static boolean success=false;
    public void actionPerformed(ActionEvent ae) {

        if(ae.getActionCommand().equals("Modeling")){

            com.mod2dgrexp.model.MOD2DGREXP_CalculateValues cv = new com.mod2dgrexp.model.MOD2DGREXP_CalculateValues();
            cv.getAnamolyValues(com.mod2dgrexp.view.MOD2DGREXP_MainPanel.captureValues());
            com.mod2dgrexp.view.MOD2DGREXP_TableView.populateEastPanel(MOD2DGREXP_Calculate
alues.obj);
```

```
            com.mod2dgrexp.view.MOD2DGREXP_MainPanel.p_East.repaint();
        }else if(ae.getActionCommand().equals("Save and Print")){
            try
            {
                MOD2DGREXP_PrintValues.printGraphValues();
            }
            catch(Exception e1) {
                e1.printStackTrace();
            }
        }else if(ae.getActionCommand().equals("Load data")){
            try {
                MOD2DGREXP_MainPanel.loadData1();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        else if(ae.getActionCommand().equals("Clear")){
            MOD2DGREXP_MainPanel.clearDefaultValues();
            MOD2DGREXP_MainPanel.clearPanel(MOD2DGREXP_MainPanel.img);
            MOD2DGREXP_TableView.populateEastPanel(rowdata);
        }
        else if(ae.getActionCommand().equals("Exit")){
            JFrame frame = null;

            int r = JOptionPane.showConfirmDialog(
                    frame,
                    "Exit MOD2DGREXP ?",
                    "Confirm Exit ",
                    JOptionPane.YES_NO_OPTION);
            if(r == JOptionPane.YES_OPTION ){
                if(success==false){
                    String fileName = MOD2DGREXP_CalculateValues.input_area_name+".jpg";
                    File f = new File(fileName);
                    f.delete();
                }
                System.exit(0);
            }

        }

    }

}
-----------------------------------------------------------------------------------------
package com.mod2dgrexp.control;

import java.io.File;
import java.io.FileWriter;
import java.text.DecimalFormat;
import javax.swing.JFileChooser;
import com.mod2dgrexp.model.MOD2DGREXP_CalculateValues;


public class MOD2DGREXP_PrintValues {

    public static void printGraphValues() throws Exception {

        try{
            String current = System.getProperty("user.dir");
            File img_file = new File(MOD2DGREXP_CalculateValues.input_area_name+".jpg");
            JFileChooser saveFile = new JFileChooser(current);
            File OutFile = saveFile.getSelectedFile();
            FileWriter myWriter = null;
            if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
            {
                OutFile = saveFile.getSelectedFile();
                if (OutFile.canWrite() || !OutFile.exists())
                {
                    File dir = new File(OutFile.getParent());
                    MOD2DGREXP_Controller.success = img_file.renameTo(new
File(dir,img_file.getName()));
                    myWriter = new FileWriter(OutFile+".html");
                    //myWriter.write("AUTOMATIC MODELING OF GRAVITY ANOMALIES OF 2D
```

```
SEDIMENTARY BASINS USING EXPONENTIAL DENSITY FUNCTION");
                        myWriter.write(" </table> </td> <td> <img src = '"+
MOD2DGREXP_CalculateValues.input_area_name +".jpg'></td></tr></table>");
                        myWriter.write("<html> <Body onLoad = \"window.print()\"><table> <tr>
<td>" +
                                "<table border = 1> <tr> <th colspan = 4>LOCATION:-
"+MOD2DGREXP_CalculateValues.input_area_name+"</th> </tr>");
                        DecimalFormat df =new DecimalFormat("0.###");
                        myWriter.write(" <tr><th colspan = 4> PROFILE NUMBER:-"+"
"+MOD2DGREXP_CalculateValues.input_profile+" </th></tr>");
                        myWriter.write(" <tr><th colspan = 4> ITERATION:-"+"
"+MOD2DGREXP_CalculateValues.o_iter+" </th></tr>");
                        myWriter.write("<tr > <th>Distance (km) </th> <th> Observed anomalies
(mGal) </th> <th> Calculated anomalies (mGal) </th> <th> Depth (km)
</th></tr>");
                        for ( int K = 1; K <= MOD2DGREXP_CalculateValues.input_n_obs; K++){
                            myWriter.write("<tr> <td>" +
MOD2DGREXP_CalculateValues.input_x_km[K]+"</td>
<td>"+df.format(MOD2DGREXP_CalculateValues.input_nob_gob[K])+"</td>
<td>"+df.format(MOD2DGREXP_CalculateValues.gc[K])+"</td>
<td>"+df.format(MOD2DGREXP_CalculateValues.z[K])+"</td></tr>");
                        }

                        myWriter.close();
                    }
                }
                else
                {
                    //pops up error message
                }
            }
            catch(Exception e1) {

                e1.printStackTrace();
            }
        }
    }
---------------------------------------------------------------------------------------------
package com.mod2dgrexp.util;

import com.mod2dgrexp.model.MOD2DGREXP_CalculateValues;

public class MOD2DGREXP_Utility {

    public static double convertDouble(String str) throws Exception {

        Double temp = new Double(str.trim());
        return temp.doubleValue();
    }

    public static String convertString(String str) throws Exception {

        String temp = new String(str.trim());
        return temp;
    }

    public static int convertInteger(String str) throws Exception {

        Integer temp = new Integer(str.trim());
        return temp.intValue();
    }

    public static int findMaximumNumber( double observe[]) {

        double max = 0.0d;
        for (int i = 0; i < observe.length; i++) {

            if (Math.abs(observe[i]) > Math.abs(max)) {

                max = observe[i];
            }
        }

        int maxVal = (int) max/3*5;

```

```
            return maxVal;
        }

        public static int findMaximumNumber( double observe) {

            double max = 0.0d;
            int maxVal=0;
            max = observe;

            if (max < 5) {

                maxVal = 5;
            }
            else if (max >= 5 && max <= 10) {

                maxVal = 10;
            }
            else if ( max > 10 && max <= 15) {

                maxVal = 15;
            }
            else if (max > 15 && max <= 20) {

                maxVal = 20;
            }
            else
            {
                maxVal = MOD2DGREXP_CalculateValues.o_iter;
            }

            return maxVal;
        }

        public static double findMaximumNumber1( double observe[]) {

            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max =Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }

        public static double[] convertDoubleArray(String str) throws Exception {

            java.util.StringTokenizer st = new java.util.StringTokenizer(str, ",");
            String temp = "";
            java.util.ArrayList arr = new java.util.ArrayList();

            while(st.hasMoreTokens()) {

                temp = st.nextToken();
                arr.add(temp);
            }
            double d_array[] = new double[arr.size() + 1];

            for(int i = 0; i <= arr.size(); i++) {

                if (i == 0)
                    d_array[i] = 0.0;
                else
                    d_array[i] = convertDouble( arr.get(i-1).toString() );
            }
            return d_array;
        }
    }
```

Annexure - 5B
Sample output



LOCATION:- Modeling - Output			
PROFILE NUMBER:- 1			
ITERATION:- 15			
Distance (km)	**Observed anamolies (mGal)**	**Calculated anamolies (mGal)**	**Depth (km)**
0.0	-1.504	-1.497	0
2.0	-6.136	-6.133	0.261
4.8	-12.331	-12.336	0.619
6.0	-15.6	-15.591	0.935
8.5	-20.99	-21.028	1.52
10.0	-23.431	-23.416	1.947
12.5	-25.885	-25.87	2.386
14.0	-26.549	-26.543	2.483
16.6	-26.634	-26.581	2.448
18.0	-26.099	-26.141	2.173
20.5	-25.193	-25.21	1.957
22.0	-24.82	-24.844	1.906
25.0	-24.412	-24.395	1.999
26.0	-24.088	-24.077	1.972
28.0	-22.908	-22.86	1.81
30.0	-20.922	-20.959	1.354
33.0	-19.76	-19.798	1.206
35.0	-20.819	-20.776	1.557
38.0	-21.265	-21.277	1.659
42.0	-19.33	-19.317	1.486
45.0	-14.505	-14.52	0.886
47.0	-10.018	-10.016	0.498
50.0	-5.558	-5.557	0.25
55.0	-2.927	-2.927	0.139
60.0	-0.499	-0.496	0

```java
package com.in2dgrexp.view;

import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import com.in2dgrexp.control.IN2DGREXP_Controller;
import com.in2dgrexp.model.IN2DGREXP_CalculateValues;


public class IN2DGREXP_MainView extends Frame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static void main(String s[])
    {
        IN2DGREXP_MainView cm = new IN2DGREXP_MainView();
        cm.setSize(1280, 768);
        cm.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                JFrame frame = null;
                int r = JOptionPane.showConfirmDialog(
                        frame,
                        "Exit IN2DGREXP ?",
                        "Confirm Exit ",
                        JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(IN2DGREXP_Controller.success==false){
                        String fileName = IN2DGREXP_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }
        });
        cm.setTitle("IN2DGREXP");
        cm.setResizable(false);
        cm.add(new IN2DGREXP_MainPanel());
        cm.setVisible(true);

    }

}
```

```java
package com.in2dgrexp.view;

import java.awt.*;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import javax.swing.JFileChooser;

import jxl.Cell;
import jxl.CellType;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;



public class IN2DGREXP_MainPanel extends Panel {

    /**
     *
     */
    private static final long serialVersionUID = 1L;


```

```
        Panel p_North, p_West,p_South;
        public static TextArea img = new TextArea(36,135);
        public static Panel p_East,p_Center;
        static TextField inputValues [] = new TextField[12];
        static TextArea graphValues = new TextArea(38,30);
        Button actionButton[] = new Button[5];
        Object rowdata[][]={};

        /**Field Area Name*/
        final static int AREA_FE = 0;
        /**Profile Name*/
        final static int PROFILE_NAME = 1;
        /**Number of observation*/
        public static final int N_OBS = 2 ;
        /**Distance(km)*/
        public static final int X_KM = 3;
        /**Number of Observed values*/
        public static final int NOB_GOB = 4;
        /** SD */
        public static final int SD_POLY = 5;
        /**LAMBDA*/
        public static final int LAMBDA_ST = 6 ;
        /**ZMIN(km)*/
        public static final int DEP_ZMIN = 7;
        /**ZMAX(km)*/
        public static final int DEP_ZMAX = 8;
        /**Number of iteration values*/
        public static final int NOB_ITER = 9;


        public IN2DGREXP_MainPanel() {

            this.setLayout(new BorderLayout());
            p_North = new Panel();
            p_West = new Panel();
            p_East = new Panel ();
            p_South = new Panel();
            p_Center = new Panel();

            Label graphLabel = new Label("INVERSION OF GRAVITY ANOMALIES OF 2D SEDIMENTARY BASINS ", Label.CENTER);
            Label graphLabel1 = new Label(" USING EXPONENTIAL DENSITY MODEL ", Label.CENTER);
            graphLabel.setFont(new Font("Arial", 10, 18));
            graphLabel1.setFont(new Font("Arial", 10, 18));
            p_Center.add(graphLabel);
            p_Center.add(graphLabel1);

            for(int i = 0; i < 11; i++){
                inputValues[i] = new TextField();
            }
            p_North.setFont(new Font("Bold",1,12));
            actionButton[0] = new Button("Load data");
            actionButton[1] = new Button("Inversion");
            actionButton[2] = new Button("Save and Print");
            actionButton[3] = new Button("Clear");
            actionButton[4] = new Button("Exit");

            this.populateNorthPanel();
            IN2DGREXP_TableView.populateEastPanel(rowdata);
            this.add(p_North, BorderLayout.NORTH);
            this.add(p_West, BorderLayout.WEST);
            this.add(p_East, BorderLayout.EAST);
            this.add(p_South, BorderLayout.SOUTH);
            p_Center.setSize(1000, 760);
            this.add(p_Center, BorderLayout.CENTER);
            img.setEditable(false);
            p_Center.add(img);
            this.setVisible(true);
        }

        public void populateNorthPanel(){
            p_North.setLayout(new GridLayout(5,6));

            p_North.add(new Label("Area Name"));
            p_North.add(inputValues[0]);
```

```
        p_North.add(new Label("Profile Name"));
        p_North.add(inputValues[1]);
        p_North.add(new Label("Number of observation:"));
        p_North.add(inputValues[2]);
        p_North.add(new Label("Distance (km)"));
        p_North.add(inputValues[3]);
        p_North.add(new Label("Observed anomalies (mGal)"));
        p_North.add(inputValues[4]);
        p_North.add(new Label("Surface density contrast (gm/cc)"));
        p_North.add(inputValues[5]);
        p_North.add(new Label("Lambda (1/km)"));
        p_North.add(inputValues[6]);
        p_North.add(new Label("Minimum depth(km):"));
        p_North.add(inputValues[7]);
        p_North.add(new Label("Maximum depth(km):"));
        p_North.add(inputValues[8]);
        p_North.add(new Label("Iterations"));
        p_North.add(inputValues[9]);
        p_North.add(new Label(""));
        p_North.add(new Label(""));
        p_North.add(new Label(""));
        p_North.add(new Label(""));

        p_North.add(actionButton[0]);
        p_North.add(actionButton[1]);
        p_North.add(actionButton[2]);
        p_North.add(actionButton[3]);
        p_North.add(actionButton[4]);

        actionButton[0].addActionListener(new com.in2dgrexp.control.IN2DGREXP_Controller()
        actionButton[1].addActionListener(new com.in2dgrexp.control.IN2DGREXP_Controller()
        actionButton[2].addActionListener(new com.in2dgrexp.control.IN2DGREXP_Controller()
        actionButton[3].addActionListener(new com.in2dgrexp.control.IN2DGREXP_Controller()
        actionButton[4].addActionListener(new com.in2dgrexp.control.IN2DGREXP_Controller()

    }

    public static HashMap captureValues(){

        HashMap h_Map = new HashMap();

        try {

            h_Map.put("N_OBS", inputValues[N_OBS].getText());
            h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
            h_Map.put("LAMBDA_ST",inputValues[LAMBDA_ST].getText());
            h_Map.put("DEP_ZMIN", inputValues[DEP_ZMIN].getText());
            h_Map.put("DEP_ZMAX", inputValues[DEP_ZMAX].getText());
            h_Map.put("X_KM", inputValues[X_KM].getText());
            h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
            h_Map.put("NOB_ITER", inputValues[NOB_ITER].getText());
            h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
            h_Map.put("PROFILE_NAME",inputValues[PROFILE_NAME].getText());
        }
        catch (Exception e) {
            e.printStackTrace();
        }

        return h_Map;

    }

    public static void clearPanel(TextArea p) {

        Graphics g = p.getGraphics();
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, 1280, 650);
    }

    public static void loadData1()throws IOException  {
        try{

            String current = System.getProperty("user.dir");
            JFileChooser chooser=new  JFileChooser(current);
            int returnVal = chooser.showOpenDialog(null);
```

```
                String dis[], gobs[];
                String disval = "" ,gobsval="";
                Workbook w;

                if(returnVal == JFileChooser.APPROVE_OPTION) {
                    File f = chooser.getSelectedFile();
                    w = Workbook.getWorkbook(f);
                    Sheet sheet = w.getSheet(0);
                    dis = new String[sheet.getRows()+1];
                    gobs = new String[sheet.getRows()+1];
                    for (int j = 0; j < sheet.getColumns(); j++) {
                        for (int i = 1; i < sheet.getRows(); i++) {
                            Cell cell = sheet.getCell(j, i);
                            CellType type = cell.getType();
                            if (type == CellType.LABEL) {
                                //  System.out.println("I got a label "
                                //     + cell.getContents());
                                IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.AREA_FE].se
Text(cell.getContents());

                            }

                            if (type == CellType.NUMBER) {
                                if (j == 1){

                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.PROFILE
NAME].setText(cell.getContents());

                                }
                                if (j == 2){

                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.N_OBS].
etText(cell.getContents());

                                }

                                if (j == 3){

                                    dis[i] = cell.getContents()+",";
                                    disval = disval + dis[i];
                                }

                                if (j == 4){

                                    gobs[i] = cell.getContents()+",";
                                    gobsval = gobsval+gobs[i];
                                }

                                if (j == 5){

                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.SD_POLY
.setText(cell.getContents());

                                }
                                if (j == 6){

                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.LAMBDA_
T].setText(cell.getContents());

                                }

                                if (j == 7){

                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.DEP_ZMI
].setText(cell.getContents());

                                }
                                if (j == 8){


                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.DEP_ZMA
].setText(cell.getContents());
                                }
                                if (j == 9){
                                    IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.NOB_ITE
```

```
].setText(cell.getContents());
                            }

                            //System.out.println("I got a number "
                            //  + cell.getContents());
                        }

                    }
                }
                IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.X_KM].setText(""+disval
;
                IN2DGREXP_MainPanel.inputValues[IN2DGREXP_MainPanel.NOB_GOB].setText(""+gob
val);
            }
        }
        catch (BiffException e) {
            e.printStackTrace();
        }



    }


    public static void clearDefaultValues() {

        inputValues[N_OBS].setText("");
        inputValues[SD_POLY].setText("");
        inputValues[LAMBDA_ST].setText("");
        inputValues[DEP_ZMIN].setText("");
        inputValues[DEP_ZMAX].setText("");
        inputValues[X_KM].setText("");
        inputValues[NOB_GOB].setText("");
        inputValues[NOB_ITER].setText("");
        inputValues[AREA_FE].setText("");
        inputValues[PROFILE_NAME].setText("");
    }

}
-------------------------------------------------------------------------------------------

package com.in2dgrexp.view;

import java.applet.Applet;
import java.awt.*;
import java.awt.Font;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.text.DecimalFormat;
import com.in2dgrexp.model.IN2DGREXP_CalculateValues;
import com.in2dgrexp.util.IN2DGREXP_Utility;


public class IN2DGREXP_DrawGraph extends Applet {
    private static final long serialVersionUID = 1L;
    float maxX,maxY;
    float maxZ;
    double inidep;
    public void drawGraph(Graphics2D g){

        g.setColor(Color.black);
        g.drawLine(150,50,150,550);
        g.drawLine(90,45 ,1040,45);

        g.drawLine(780, 45, 780, 560);
        g.drawLine(90, 560, 1040, 560);
        g.drawLine(1040, 45, 1040, 560);
        g.drawLine(90, 45, 90, 560);

        g.drawString("DISTANCE(Km)",315 ,295);
        String a[]={"A","N","O","M","A","L","Y","(m","G","a","l","s)"};
        String b[]={"D","E","P","T","H","(k","m)"};
        for(int i=0;i<a.length;i++){

```

```
                g.drawString(""+a[i],100 ,60+(i*20));
            }
            for(int i=0;i<b.length;i++){

                g.drawString(""+b[i],100 ,350+(i*20));
            }
        }

        public void plot(Graphics2D g){

            maxX = (float)
    IN2DGREXP_Utility.findMaximumNumber1(IN2DGREXP_CalculateValues.input_x_km);
    //i_no_obs;//
            maxY = IN2DGREXP_Utility.findMaximumNumber(IN2DGREXP_CalculateValues.input_nob_gob)
            inidep  = IN2DGREXP_Utility.findMaximumNumber1(IN2DGREXP_CalculateValues.z);
            maxZ = (float) inidep;
            DecimalFormat df = new DecimalFormat("0.#");
            DecimalFormat df1= new DecimalFormat("0.##");
            g.drawString("0",140,60);
            g.drawString(""+(int)maxY  ,125,300);
            g.drawString("0", 125,310);
            g.drawString("|", 600, 308);
            g.drawString(""+df.format(IN2DGREXP_CalculateValues.input_x_km[IN2DGREXP_CalculateV
    lues.input_n_obs]) ,600,320);
            g.draw(new Line2D.Float(150, 300,600,300));

            float points = maxX/5;
            int yInterval=50;
            int zInterval=50;
            float xInterval=(float)
    (IN2DGREXP_CalculateValues.input_x_km[IN2DGREXP_CalculateValues.input_n_obs]/5);
            float xplot=0;
            for (float x = xInterval, j =1; x < 600; x+=xInterval){

                xplot=xplot+xInterval;
                if(j>4)
                    break;
                g.drawString("|",(float) (150+(450*x/maxX)), 308);
                g.drawString("" + df.format(xplot), (float) (150+(450*x/maxX))-3, 323);
                j++;
            }
            points = maxY/5;
            for (int x = yInterval, j =1; x < 250; x+=yInterval){

                g.drawString("-",148,50+x);
                g.drawString("" +(int)(points*j), 125,50+x);
                j++;
            }
            float point =maxZ/5 ;
            for(int x = zInterval+250,j =1; x < 550; x+=zInterval){

                if(j>4)
                    break;
                g.drawString("-",148,50+x);
                g.drawString("" +df.format(point*j),125,50+x);
                j++;
            }
            g.drawString("-",148,552);
            g.drawString(""+df1.format(maxZ), 125, 550);
        }

        public void plotXYCoordinates (Graphics2D g){

            float prevx = 150;
            float prevy =50;
            float xpoint=0;
            float ypoint=0;
            float gypoint=0;

            for (int k =1; k <= IN2DGREXP_CalculateValues.input_n_obs; k++){

                xpoint = (float)(450 * IN2DGREXP_CalculateValues.input_x_km[k]/ maxX) ;
                ypoint = (float)(250 * IN2DGREXP_CalculateValues.gc[k]/ maxY);
                gypoint = (float)(250 * IN2DGREXP_CalculateValues.input_nob_gob[k]/ maxY);
```

```
            g.setColor(Color.BLACK);
            g.draw(new Line2D.Float(prevx, prevy,150+xpoint,50+ypoint));
            g.setColor(Color.BLUE);
            g.setFont(new Font("Arial", 20, 55));
            g.drawString(".",150+xpoint-6 ,50+gypoint);

            prevx = 150+xpoint;
            prevy = 50+ypoint;

        }
    }

    public void plotZCoordinates (Graphics2D g){

        float prevx = 150;
        float prevz =300;
        float xpoint=0;
        float zpoint=0;

        GradientPaint gradient = new GradientPaint(10, 10, Color.yellow, 30, 200, Color.MAGENTA, true);
        g.setPaint(gradient);
        g.fill(new Rectangle2D.Float(151 ,300,450, 250 ));
        for (int k = 1; k <= IN2DGREXP_CalculateValues.input_n_obs; k++){

            xpoint = (float)(450 * IN2DGREXP_CalculateValues.input_x_km[k]/ maxX);
            zpoint = (float)(250 * IN2DGREXP_CalculateValues.z[k]/maxZ );
            float vary=prevx;

            g.setColor(Color.red);
            if(prevz<=300+zpoint){
                while(vary<= 150+xpoint){
                    g.draw(new Line2D.Float(prevx, prevz,vary,300+zpoint));
                    vary = (float) (vary+0.001);
                }
                g.fill(new Rectangle2D.Float(prevx ,300+zpoint,(150+xpoint)-prevx, 250-zpoint ));
            }
            else{

                vary = prevz;
                while( 300 + zpoint <= vary){
                    g.draw(new Line2D.Float(prevx, prevz,150+xpoint,vary));
                    vary = (float) (vary-0.001);
                }
                g.fill(new Rectangle2D.Float(prevx ,prevz,(150+xpoint)-prevx, 550-prevz ));
            }
            g.setColor(Color.black);
            g.draw(new Line2D.Float(prevx, prevz,150+xpoint,300+zpoint));

            prevx = 150+xpoint;
            prevz = 300+zpoint;

        }
        g.setColor(Color.white);
        g.fill(new Rectangle2D.Float(151 ,550,450, 50 ));
    }

    public void drawOBJ(Graphics2D g2) {

        g2.setColor(Color.BLACK);

        g2.drawLine(820, 70, 820, 160);
        g2.drawLine(820, 160, 910, 160);
        g2.drawString("J", 800, 90);

        double maxOb = IN2DGREXP_Utility.findMaximumNumber1(IN2DGREXP_CalculateValues.o_funct);
        int ini = IN2DGREXP_Utility.findMaximumNumber(IN2DGREXP_CalculateValues.o_iter);
        if(ini==5)
            ini= ini+1;
        int maxiter = ( ini / 3 * 5 ) * 2;
        int point;
        int xInterval = 22;
```

```java
            point = ( ( ini ) / 3 * 5 ) / 5;

            for (int x = xInterval, j = 1; x < 90 ; x += xInterval) {

                g2.drawString("'", 821+x, 170);
                g2.drawString("" + (point*j), 820 + x-3, 175);
                j++;
            }

            float prevx = 820;
            float prevy = 70;
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= IN2DGREXP_CalculateValues.o_iter; i++) {

                xpoint = (float)( 250 * i /maxiter );
                ypoint = 70 - (float)  ( ( 90 * (IN2DGREXP_CalculateValues.o_funct[i]) / maxOb
);

                if(i==IN2DGREXP_CalculateValues.o_iter){
                    g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint-4, 90 + ypoint));
                }
                else {
                    g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 90 + ypoint));
                }
                prevx = 820 + xpoint;
                prevy = 90 + ypoint;

            }
            DecimalFormat d1= new DecimalFormat("0.###");
            DecimalFormat d= new DecimalFormat("0.#");
            g2.drawString(" "+d.format(IN2DGREXP_CalculateValues.o_funct[1]), 780, 70);
            g2.drawString("
"+d1.format(IN2DGREXP_CalculateValues.o_funct[IN2DGREXP_CalculateValues.o_iter]),
820 + xpoint, 90 + ypoint);
            g2.setFont(new Font("Arial", 40,11));
            g2.drawString ("Iterations",850,186);
        }

        public void drawSd(Graphics2D g2) {
            g2.setColor(Color.black);
            g2.drawLine(780, 200, 1040, 200);
            g2.setColor(Color.red);
            g2.setFont(new Font("Arial", 20, 12));
            DecimalFormat d= new DecimalFormat("0.##");
            DecimalFormat d2= new DecimalFormat("0.#");
            DecimalFormat d1= new DecimalFormat("0.###");

            g2.drawString(""+d.format(inidep),790,550);
            g2.drawString("-",820,552);
            g2.drawString("0",807 ,300);
            g2.drawLine(820, 300, 910, 300);
            g2.draw(new Line2D.Float(820, 300, 820,  550));

            double maxOb1 = IN2DGREXP_Utility.findMaximumNumber1(IN2DGREXP_CalculateValues.vsd)
            float points =maxZ/5 ;
            int zInterval=50;
            for(int x = zInterval+250,j =1; x < 550; x+=zInterval){

                if(j>4)
                    break;
                g2.drawString("-",820,50+x+2);
                g2.drawString("" +d2.format(points*j),790,50+x);
                j++;

            }

            float prevx = 820+ (float) ( ( 90 * ( Math.abs(IN2DGREXP_CalculateValues.vsd[1] ))
maxOb1 ) );
            float prevy = 300;
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= IN2DGREXP_CalculateValues.count; i++) {
```

```java

                    xpoint = (float)( 90 * Math.abs(IN2DGREXP_CalculateValues.vsd[i]) / maxOb1 );
                    ypoint = (float)( 250 * IN2DGREXP_CalculateValues.dep[i] / maxZ );

                    g2.setColor(Color.blue);
                    g2.draw(new Line2D.Float(prevx, prevy, 820 + xpoint, 300 + ypoint));

                    prevx = 820 + xpoint;
                    prevy = 300 + ypoint;
                }

                g2.drawString(""+d.format(IN2DGREXP_CalculateValues.vsd[1] ),805+ (float) ( ( 90 * Math.abs(IN2DGREXP_CalculateValues.vsd[1] )) / maxOb1 ) ) ,300 );
                g2.drawString(""+d1.format(IN2DGREXP_CalculateValues.vsd[IN2DGREXP_CalculateValues.count] ),820+ (float) ( ( 90 * ( Math.abs(IN2DGREXP_CalculateValues.vsd[IN2DGREXP_CalculateValues.count] )) / maxOb1 ) ), 300+(float)( 250 * inidep / maxZ ) );
                g2.setColor(Color.BLACK);
                g2.drawString("Variation of density contrast " , 800,220);
                g2.drawString("with depth" , 850,240);
                g2.setFont(new Font("Arial", 40,11));
                g2.drawString ("Density contrast",830,285);
                g2.drawString ("(gm/cc)",843,295);
                g2.drawString("Z(km)", 790,(float)( 300+((250*inidep/maxZ))/2));

            }

            /** Index of Graph*/
            public void idex(Graphics2D g){

                g.setColor(Color.BLUE);
                g.setFont(new Font("Arial", 20, 50));
                g.drawString(" ...     ",595,70);
                g.setFont(new Font("Arial", 20, 12));
                g.drawString("Observed anomalies",650,70);
                g.setColor(Color.BLACK);
                g.drawString("____:",615,87);
                g.drawString("Calculated anomalies",650,90);
                g.setColor(Color.black);
                g.drawString("<--------------------------------------------------:",450,340);
                g.setColor(Color.RED);
                g.drawString("Estimated Depth ",660,340);
                g.drawString(" Structure",660,355);
            }

        }

        ---------------------------------------------------------------------------------------------

        package com.in2dgrexp.view;

        import java.awt.Dimension;

        import javax.swing.JScrollPane;
        import javax.swing.JTable;

        public class IN2DGREXP_TableView {

            public static  void populateEastPanel(Object rowData[][]) {
                com.in2dgrexp.view.IN2DGREXP_MainPanel.p_East.removeAll();


                Object columnNames[] = {"Distance(km)", "Observed anamolies (mGal)", "Calculated anamolies (mGal)","Depth(km)","ERROR(mGal)"};


                JTable table = new JTable(rowData, columnNames);

                table.setPreferredScrollableViewportSize(new Dimension(300,550));


                JScrollPane scrollPane = new JScrollPane(table);
                scrollPane.setAutoscrolls(true);


```

```
            com.in2dgrexp.view.IN2DGREXP_MainPanel.p_East.add(scrollPane);
            com.in2dgrexp.view.IN2DGREXP_MainPanel.p_East.validate();

            com.in2dgrexp.view.IN2DGREXP_MainPanel.p_East.setVisible(true);

            //System.out.println(p_East.isVisible());
        }

    }
    ------------------------------------------------------------------------------------------

    package com.in2dgrexp.model;

    import java.awt.Color;
    import java.awt.Graphics;
    import java.awt.Graphics2D;
    import java.awt.event.MouseAdapter;
    import java.awt.event.MouseEvent;
    import java.awt.event.MouseListener;
    import java.awt.image.BufferedImage;
    import java.io.File;
    import java.io.FileOutputStream;
    import java.text.DecimalFormat;
    import java.util.HashMap;

    import javax.imageio.ImageIO;

    import com.in2dgrexp.util.IN2DGREXP_Utility;
    import com.in2dgrexp.view.IN2DGREXP_DrawGraph;
    import com.in2dgrexp.view.IN2DGREXP_MainPanel;


    public class IN2DGREXP_CalculateValues  {

        public static double []o_funct;
        public static int o_iter,count;
        public static int input_n_obs ;
        public static double funct1,lambda,funct2;
        public static double input_x_km[];
        public static double input_nob_gob[] ;
        public static double []gc;
        public static double []z;
        public static double []err;
        public static double []vsd = null;
        public static double []dep = null;
        public static String input_area_name = "";
        public static String input_profile = "";
        static BufferedImage image;
        public static Object obj[][] = null;

        public void getAnamolyValues(HashMap h_Map) {


            double GK = 13.3333;
            double PI = 22.0 / 7.0;
            double input_sd_poly = 0;
            double input_lambda_val = 0;
            double input_zmin_km = 0;
            double input_zmax_km = 0;
            int input_nob_iter=0;

            try {
                input_n_obs =  IN2DGREXP_Utility.convertInteger((String)h_Map.get("N_OBS"));
                input_sd_poly = IN2DGREXP_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                input_lambda_val =
    IN2DGREXP_Utility.convertDouble((String)h_Map.get("LAMBDA_ST"));
                input_zmin_km = IN2DGREXP_Utility.convertDouble((String)h_Map.get("DEP_ZMIN"));
                input_zmax_km = IN2DGREXP_Utility.convertDouble((String)h_Map.get("DEP_ZMAX"));
                input_x_km = IN2DGREXP_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
                input_nob_gob =
    IN2DGREXP_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                input_nob_iter =
    IN2DGREXP_Utility.convertInteger((String)h_Map.get("NOB_ITER"));
                input_area_name = IN2DGREXP_Utility.convertString((String)h_Map.get("AREA_FE"))
```

```
                input_profile = IN2DGREXP_Utility.convertString((String)h_Map.get("PROFILE_NAME"));
            }
            catch (Exception e) {
                e.printStackTrace();
            }

            int np = input_n_obs - 2;
            int np1 = np + 1;

            double ALERR = 0.001*input_n_obs;
            o_funct = new double[input_nob_iter+1];
            gc = new double[input_n_obs +1];

            double []b = new double[input_n_obs +1];
            err = new double[input_n_obs +1];
            double []par = new double[input_n_obs +2];
            double []par1 = new double[input_n_obs +2];
            double []par2 = new double[input_n_obs +2];

            z = new double[input_n_obs +2];

            double []g1 = new double[input_n_obs+1];
            double []g2 = new double[input_n_obs+1];
            //double err[]=new double[input_n_obs+1];
            double [][]s = new double[input_n_obs+1][input_n_obs+1];
            double []dupar = new double[input_n_obs+1];
            double [][]p1 = new double[input_n_obs+1][input_n_obs+1];

            double DPAR = 0.1;
            int stn=0;

            lambda = 0.5;

            for (int K = 2; K <= input_n_obs - 1; K++) {

                stn=stn+1;
                if(input_lambda_val==0)
                    par[stn]= input_nob_gob[K]/(13.3333*PI*input_sd_poly);
                else
                    par[stn] = -(1 / input_lambda_val) * Math.log(1 - ((input_lambda_val * input_nob_gob[K]) / (GK * PI * input_sd_poly)));

            }

            GBSN(input_n_obs,input_x_km,par,input_sd_poly,input_lambda_val,GK,gc);
            funct1 = 0.0;
            for (int K = 1; K <= input_n_obs; K++) {

                err[K] = input_nob_gob[K] - gc[K];
                funct1 = funct1 + Math.pow(err[K], 2);
            }
            funct1=Math.sqrt(funct1/input_n_obs);

            int iter ;

            for (iter = 1; iter <= input_nob_iter; iter++) {

                o_funct[iter]=funct1;
                o_iter = iter;

                z[1]=0.0001;
                z[input_n_obs]=0.0001;
                int ist =0;
                for (int KK = 2; KK <= input_n_obs-1; KK++) {

                    ist =ist+1;
                    z[KK] = par[ist];

                }

                for (int k = 1; k <= np; k++) {
                    par1[k] =par[k];

```

```
}
for (int i = 1; i <= np; i++) {
    par1[i] = par[i]+DPAR/2;
    GBSN(input_n_obs,input_x_km,par1,input_sd_poly,input_lambda_val,GK,g1);
    par1[i] = par[i]-DPAR/2;
    GBSN(input_n_obs,input_x_km,par1,input_sd_poly,input_lambda_val,GK,g2);

    for (int k = 1; k <= input_n_obs; k++) {

        s[i][k] = (g1[k]-g2[k])/DPAR;

    }
}
for (int j = 1; j <= np1; j++) {

    for (int I = 1; I <= np; I++) {

        p1[I][j] = 0;
    }
}
for ( int J = 1; J <= np; J++) {

    for ( int I = 1; I <= np; I++) {

        for ( int K = 1; K <= input_n_obs; K++) {

            p1[I][J] = p1[I][J] + s[I][K] * s[J][K];

        }
    }
}

for ( int J = 1; J <= np; J++) {

    for ( int K = 1; K <= input_n_obs; K++) {

        p1[J][np1] = p1[J][np1] + err[K] * s[J][K];

    }
}

do {

    double CON  = (lambda + 1.0);

    for(int i=1;i<=np;i++){
        dupar[i] = par[i];
    }
    for (int L = 1; L <= np; L++) {

        for (int J = 1; J <= np; J++) {

            if (L - J == 0) {

                p1[L][J] = p1[L][J] * CON;

            }
        }
    }

    double KS [] = new double[2];

    SIMEQ(p1,b,np,KS);
    for (int I = 1; I <= np; I++) {

        par2[I] = dupar[I] +0.5* b[I];

        if (par2[I] <= input_zmin_km)
            par2[I] = input_zmin_km;

        if (par2[I] > input_zmax_km)
            par2[I] = input_zmax_km;
```

```java
                    }
                    GBSN(input_n_obs,input_x_km,par2,input_sd_poly,input_lambda_val,GK,gc);

                    funct2 = 0;

                    for (int K = 1; K <= input_n_obs; K++) {

                        err[K] = input_nob_gob[K] - gc[K];
                        funct2 = funct2 + Math.pow(err[K], 2);

                    }
                    funct2 = Math.sqrt(funct2/input_n_obs);

                    if (funct1 - funct2 < 0) {

                        if (lambda - 13.0 > 0) {

                            break;
                        }
                    }

                    if (funct1 - funct2 < 0) {

                        if (lambda - 13.0 <= 0) {

                            lambda = lambda * 2.0;

                            for (int I = 1; I <= np; I++) {

                                for (int J = 1; J <= np; J++) {

                                    if (I - J == 0) {

                                        p1[I][J] = p1[I][J] / CON;

                                    }
                                }
                            }
                        }
                    }

                } while (funct1 <= funct2);

                setGraphValues(input_n_obs,o_iter, input_x_km, z, input_nob_gob, gc, err,
        funct1,lambda,  input_area_name);
                denCal(input_sd_poly,input_lambda_val);
                drawGraph();
                for(int I = 2; I <= input_n_obs-1; I++) {
                    z[I]=par[I-1];
                }
                funct1 = funct2;
                lambda = (lambda / 2.0);
                for(int I = 1; I <= input_n_obs; I++) {

                    par[I] = par2[I];


                }

                if (funct2 - ALERR <= 0)
                    break;

            }
        }


        public static void denCal(double sd,double la){

            int i = 1;
            double z1= 0.0001;
            double z2 = IN2DGREXP_Utility .findMaximumNumber1(IN2DGREXP_CalculateValues.z);
            vsd = new double[(int) Math.pow(input_n_obs,2)];
            dep = new double[(int) Math.pow(input_n_obs,2)];
            while(z1 <= z2){

```

```java
                double dc =  sd*Math.exp(-la*z1);
                vsd[i] = dc;
                dep[i] = z1;

                z1 = z1+0.1;
                i++;
            }

            count = i;
            vsd[count] = sd*Math.exp(-la*z2);
            dep[count] = z2;

        }



        public double[] GBSN(int n,double []x,double []zvv,double sd,double la,double gk,double
    []gc){

            double ggc=0;
            double []xx = new double [n+2];
            double []zv = new double [n+2];
            double []zt = new double[10000];
            double []x1 = new double[10000];
            double []gs = new double[10000];

            zv[1] = 0.0001;
            zv[n] = 0.0001;
            int itn =0;
            for(int jl =2;jl<=n-1;jl++){
                itn = itn+1;
                zv[jl] = zvv[itn];
            }
            for(int JJ = 1 ; JJ <= n ;JJ++){
                gc[JJ]=0.0;
            }
            for(int k1=1;k1<=n;k1++){
                for(int k2=1;k2<=n;k2++){
                    xx[k2]=x[k2]-x[k1];
                }
                xx[n+1]=xx[1];
                zv[n+1]=zv[1];
                double grav = 0;
                for(int i=1;i<=n;i++){
                    double dxx = xx[i+1] - xx[i];
                    double dzz = zv[i+1] - zv[i];
                    double r = Math.sqrt(Math.pow(dxx,2)+Math.pow(dzz,2));
                    double c = dxx / r;
                    double s = dzz / r;
                    double ct = c / s;
                    double dx = (x[2]-x[1])/4;
                    double ZB = Math.abs(zv[i+1]-zv[i]);
                    int nd = (int)(ZB/dx)+1;
                    double n1 = nd / 2;
                    if (nd - (2 * n1 ) < 0 || nd - ( 2 * n1 ) > 0) {
                        nd = nd + 1;
                    }
                    double DZ = ZB / nd;
                    int N2 = nd + 1;

                    if(zv[i+1]-zv[i]==0)
                        break;
                    for(int JZ=1;JZ<=N2;JZ++){
                        if(zv[i+1]-zv[i]<0) {
                            zt[JZ] = zv[i] - DZ * (JZ-1);
                        }
                        if(zv[i+1]-zv[i]>0)     {
                            zt[JZ] = zv[i] + DZ * (JZ-1);
                        }

                        if(zt[JZ]<0)
                            zt[JZ] = 0;
                        x1[JZ] = xx[i]+(zt[JZ]-zv[i]) * ct;
                        if(Math.abs(x1[JZ])<0.01)
                            x1[JZ] = 0;
```

```
                    }

                    for(int JZ=1;JZ<=N2;JZ++){
                        double DC = (sd * Math.exp(-la * zt[JZ]));
                        double a = xx[i] - zv[i] * ct;
                        double anum = a + zt[JZ] * ct;
                        gs[JZ]= -13.3333*DC*Math.atan(anum / zt[JZ]);

                    }
                    ggc=SIMP(gs,zt,N2,ggc);
                    grav = grav + ggc;

                }

                gc[k1] = grav;


            }
            return gc;
        }

        public double SIMP(double []gs,double []z,int n,double ggc) {

            double dz = z[2]-z[1];
            double sum1 = 0.0;
            double sum2 = 0.0;
            int n1 = n / 2;
            int n4 = n1 - 1;
            for(int I = 1; I <= n1; I++) {
                int n2 = 2 * I;
                sum1 = sum1 + gs[n2];
            }
            for(int I = 1; I <= n4; I++) {
                int n3 = 2 * I +1;
                sum2 = sum2 + gs[n3];
            }
            ggc = gs[1]+4*sum1+2*sum2+gs[n];
            ggc = ggc * dz / 3.0;
            return ggc;
        }
        public static double []SIMEQ(double p[][], double b[], int n, double KS[]) {

            int I = n + 1;
            double []a = new double[n*n+1];

            for (int I1 = 1; I1 <= n; I1++) {

                for (int I2 = 1; I2 <= n; I2++) {

                    int I3 = (I1 - 1) * n + I2;
                    a[I3] = p[I2][I1];

                }
            }

            for (int I4 = 1;I4 <= n; I4++) {

                b[I4] = p[I4][I];

            }
            double TOL = 0;

            KS[0] = 0;
            int JJ = - n;
            int IT;
            int NY = 0;
            for (int J = 1;J <= n; J++) {

                int JY = J + 1;
                JJ = JJ + n + 1;
                double biga = 0;

                IT = JJ - J;
                int imax = 0;
```

```java
                for (int i = J; i <= n; i++) {

                    int IJ = IT + i;
                    if (Math.abs(biga) - Math.abs(a[IJ]) < 0) {

                        biga = a[IJ];
                        imax = i;
                    }

                }
                int I1 = 0;

                if (Math.abs(biga) - TOL <= 0) {

                    KS[1] = 1;
                    return KS;
                }

                else {

                    I1 = J + n * (J - 2);
                    IT = imax - J;

                }


                double save;
                for (int K = J;K <= n; K++) {

                    I1 = I1 + n;
                    int I2 = I1 + IT;
                    save = a[I1];
                    a[I1] = a[I2];
                    a[I2] = save;
                    a[I1] = a[I1] / biga;
                }

                save = b[imax];
                b[imax] = b[J];
                b[J] = save / biga;
                int IQS = 0;

                if (J - n < 0 || J - n > 0) {

                    IQS = n * (J - 1);
                    for (int IX = JY;IX <= n; IX++) {

                        int IXJ = IQS + IX;
                        IT = J - IX;
                        for (int JX = JY;JX <= n; JX++) {

                            int IXJX = n * (JX - 1) + IX;
                            int JJX = IXJX + IT;

                            a[IXJX] = a[IXJX] - (a[IXJ] * a[JJX]);
                        }

                        b[IX] = b[IX] - (b[J] * a[IXJ]);
                    }

                }
            }
            NY = n - 1;
            IT = n * n;


            for (int J = 1;J <= NY; J++) {

                int ia = IT - J;
                int ib = n - J;
                int ic = n;
                for (int K = 1;K <= J; K++) {

                    b[ib] = b[ib] - a[ia] * b[ic];
                    ia = ia - n;
```

```
                ic = ic - 1;

            }
        }
        return b;
    }

    public static void  setGraphValues(int i_no_obs, int ite, double []dis, double []dep, double []GOBS, double []GCAL, double ERR[], double FUNCT, double lamb,String Area_fe) {

        obj = new Object[i_no_obs+21][5];

        DecimalFormat df =new DecimalFormat("0.###");
        DecimalFormat d1 =new DecimalFormat("0.#######");
        for(int K=1;K<=i_no_obs;K++){

            obj[K][0]= "" + dis[K];
            obj[K][1]= "" + df.format(GOBS[K]);
            obj[K][2]= "" + df.format(GCAL[K]);
            obj[K][3]= "" + df.format(dep[K]);
            obj[K][4]= "" + df.format(ERR[K]);
        }

        obj[0][0] ="ITERATION";
        obj[0][1] = "=" +" "+ite;

        obj[i_no_obs+2][0] = "OBJECTIVE " ;
        obj[i_no_obs+2][1] = "FUNCTION =";
        obj[i_no_obs+2][2] = d1.format(FUNCT);

    }

    public static void drawGraph(){
        final IN2DGREXP_DrawGraph dg = new IN2DGREXP_DrawGraph();
        try{
            int width = 1280;
            int height = 650;
            BufferedImage buffer = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
            Graphics g1= buffer.createGraphics();
            g1.setColor(Color.WHITE);
            g1.fillRect(0,0,width,height);
            Graphics2D g2 = (Graphics2D)g1 ;
            dg.plot(g2);
            dg.plotXYCoordinates(g2);
            dg.plotZCoordinates(g2);
            dg.idex(g2);
            dg.drawGraph(g2);
            dg.plot(g2);
            dg.drawOBJ(g2);
            dg.drawSd(g2);
            FileOutputStream os = new FileOutputStream( IN2DGREXP_CalculateValues.input_area_name+".jpg");
            ImageIO.write(buffer, "jpg", os);
            os.close();

            String path = IN2DGREXP_CalculateValues.input_area_name+".jpg";
            image = ImageIO.read(new File(path));

            Graphics g_image = IN2DGREXP_MainPanel.img.getGraphics();
            g_image.drawImage(image, -80, -40, image.getWidth(), image.getHeight(), dg);
            MouseListener ml3 = new MouseAdapter(){
                public void mouseClicked(MouseEvent e){
                    Graphics g_image = IN2DGREXP_MainPanel.img.getGraphics();
                    g_image.drawImage(image, -80,-40,image.getWidth(), image.getHeight(),dg);
                }
            };
            IN2DGREXP_MainPanel.img.addMouseListener(ml3);
        }
        catch (Exception e2) {

            e2.printStackTrace();
        }
    }
```

```
}
--------------------------------------------------------------------------------------------

package com.in2dgrexp.control;

import java.awt.event.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import javax.swing.*;
import com.in2dgrexp.model.IN2DGREXP_CalculateValues;
import com.in2dgrexp.view.IN2DGREXP_DrawGraph;
import com.in2dgrexp.view.IN2DGREXP_MainPanel;


public class IN2DGREXP_Controller implements ActionListener {
    IN2DGREXP_DrawGraph dg = new IN2DGREXP_DrawGraph();
    public static boolean success=false;
    Object rowdata[][]={};
    public void actionPerformed(ActionEvent ae) {

        if(ae.getActionCommand().equals("Inversion")){

            com.in2dgrexp.model.IN2DGREXP_CalculateValues cv = new
com.in2dgrexp.model.IN2DGREXP_CalculateValues();
            cv.getAnamolyValues(com.in2dgrexp.view.IN2DGREXP_MainPanel.captureValues());
            com.in2dgrexp.view.IN2DGREXP_TableView.populateEastPanel(IN2DGREXP_CalculateVal
es.obj);
            com.in2dgrexp.view.IN2DGREXP_MainPanel.p_East.repaint();

        }else if(ae.getActionCommand().equals("Save and Print")){
            try {
                String current = System.getProperty("user.dir");
                File img_file = new File(IN2DGREXP_CalculateValues.input_area_name
+".jpg");
                JFileChooser saveFile = new JFileChooser(current);
                File OutFile = saveFile.getSelectedFile();
                FileWriter myWriter = null;
                if(saveFile.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
                {
                    OutFile = saveFile.getSelectedFile();

                    if (OutFile.canWrite() || !OutFile.exists())
                    {
                        File dir = new File(OutFile.getParent());
                        success = img_file.renameTo(new File(dir,img_file.getName()));
                        myWriter = new FileWriter(OutFile+".html");
                        myWriter.write(" </table> </td> <td> <img src = '"+
IN2DGREXP_CalculateValues.input_area_name
+".jpg'></td></tr></table>");
                        myWriter.write("<html> <Body onLoad = \"window.print()\"><table>
<tr> <td>" +
                                "<table border = 1> <tr> <th colspan = 4>LOCATION:-
"+IN2DGREXP_CalculateValues.input_area_name+"</th> </tr>");

                        DecimalFormat df =new DecimalFormat("0.###");
                        myWriter.write(" <tr><th colspan = 4> PROFILE NUMBER:-"+"
"+IN2DGREXP_CalculateValues.input_profile+" </th></tr>");
                        myWriter.write(" <tr><th colspan = 4> ITERATION:-"+"
"+IN2DGREXP_CalculateValues.o_iter+" </th></tr>");
                        myWriter.write("<tr > <th>Distance (km) </th> <th> Observed
anomalies (mGal) </th> <th> Calculated anomalies (mGal) </th> <th>
Depth (km) </th><th> ERROR (mGal) </th></tr>");
                        for ( int K = 1; K <= IN2DGREXP_CalculateValues.input_n_obs; K++){

                            myWriter.write("<tr> <td>" +
IN2DGREXP_CalculateValues.input_x_km[K]+"</td>
<td>"+df.format(IN2DGREXP_CalculateValues.input_nob_gob[K])+"</t
d> <td>"+df.format(IN2DGREXP_CalculateValues.gc[K])+"</td>
<td>"+df.format(IN2DGREXP_CalculateValues.z[K])+"</td><td>"+df.f
ormat(IN2DGREXP_CalculateValues.err[K])+"</td></tr>");
                        }

```

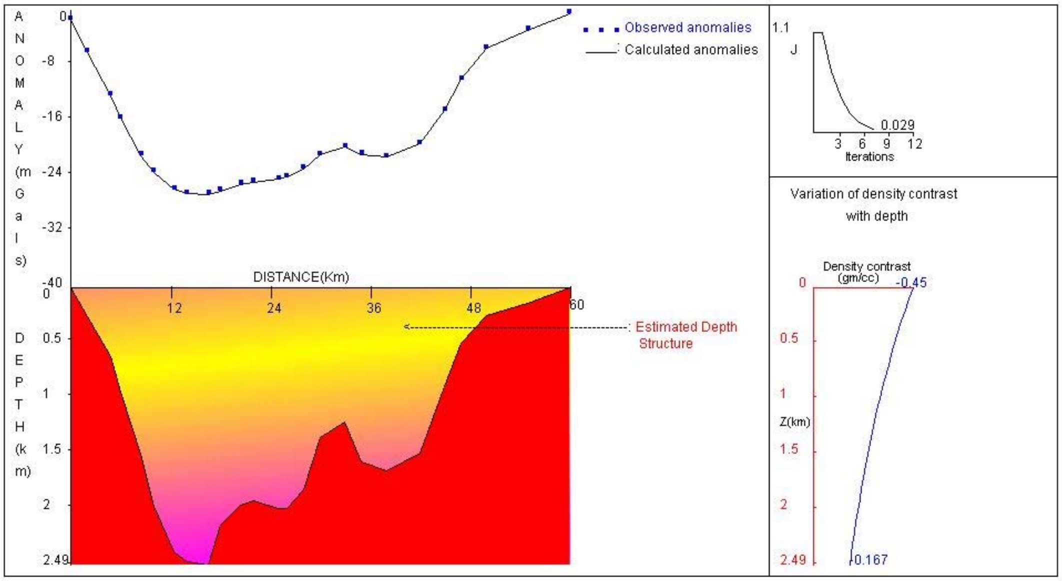
```java
                    }
                    myWriter.close();
                }
            }
            catch (Exception e) {

                e.printStackTrace();
            }

        }
        else if(ae.getActionCommand().equals("Load data")){
            try {
                IN2DGREXP_MainPanel.loadData1();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        else if(ae.getActionCommand().equals("Clear")){
            IN2DGREXP_MainPanel.clearDefaultValues();
            IN2DGREXP_MainPanel.clearPanel(IN2DGREXP_MainPanel.img);
            com.in2dgrexp.view.IN2DGREXP_TableView.populateEastPanel(rowdata);
        }

        else if(ae.getActionCommand().equals("Exit")){
            JFrame frame = null;

            int r = JOptionPane.showConfirmDialog(
                    frame,
                    "Exit IN2DGREXP ?",
                    "Confirm Exit ",
                    JOptionPane.YES_NO_OPTION);
            if(r == JOptionPane.YES_OPTION ){
                if(success==false){
                    String fileName = IN2DGREXP_CalculateValues.input_area_name+".jpg";
                    File f = new File(fileName);
                    f.delete();
                }
                System.exit(0);
            }
        }

    }

}
```

```java
package com.in2dgrexp.util;

import com.in2dgrexp.model.IN2DGREXP_CalculateValues;

public class IN2DGREXP_Utility {

    public static double convertDouble(String str) throws Exception {

        Double temp = new Double(str.trim());
        return temp.doubleValue();
    }

    public static String convertString(String str) throws Exception {

        String temp = new String(str.trim());
        return temp;
    }

    public static int convertInteger(String str) throws Exception {

        Integer temp = new Integer(str.trim());
        return temp.intValue();
    }

    public static int findMaximumNumber( double observe[]) {

```

```java
            double max = 0.0d;
            for (int i = 0; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max = observe[i];
                }
            }

            int maxVal = (int) max/3*5;
            return maxVal;
        }

        public static int findMaximumNumber( double observe) {

            double max = 0.0d;
            int maxVal=0;
            max = observe;

            if (max < 5) {

                maxVal = 5;
            }
            else if (max >= 5 && max <= 10) {

                maxVal = 10;
            }
            else if ( max > 10 && max <= 15) {

                maxVal = 15;
            }
            else if (max > 15 && max <= 20) {

                maxVal = 20;
            }
            else
            {
                maxVal = IN2DGREXP_CalculateValues.o_iter;
            }
            return maxVal;
        }

        public static double findMaximumNumber1( double observe[]) {

            double max = 0.0d;
            for (int i = 1; i < observe.length; i++) {

                if (Math.abs(observe[i]) > Math.abs(max)) {

                    max =Math.abs(observe[i]);
                }
            }

            double maxVal =  max;
            return maxVal;
        }

        public static double[] convertDoubleArray(String str) throws Exception {

            java.util.StringTokenizer st = new java.util.StringTokenizer(str, ",");
            String temp = "";
            java.util.ArrayList arr = new java.util.ArrayList();

            while(st.hasMoreTokens()) {

                temp = st.nextToken();
                arr.add(temp);
            }
            double d_array[] = new double[arr.size() + 1];

            for(int i = 0; i <= arr.size(); i++) {

                if (i == 0)
                    d_array[i] = 0.0;
                else
```

```
                d_array[i] = convertDouble( arr.get(i-1).toString() );
            }
            return d_array;
        }
    }
```

Annexure - 6B
Sample output



LOCATION:- Inversion-Output				
PROFILE NUMBER:- 1				
ITERATION:- 7				
Distance (km)	**Observed anamolies (mGal)**	**Calculated anamolies (mGal)**	**Depth (km)**	**ERROR (mGal)**
0.0	-1.504	-1.5	0	-0.004
2.0	-6.136	-6.154	0.264	0.018
4.8	-12.331	-12.347	0.621	0.016
6.0	-15.6	-15.602	0.943	0.002
8.5	-20.99	-20.985	1.5	-0.005
10.0	-23.431	-23.41	1.961	-0.021
12.5	-25.885	-25.856	2.366	-0.029
14.0	-26.549	-26.536	2.465	-0.013
16.6	-26.634	-26.592	2.485	-0.042
18.0	-26.099	-26.104	2.136	0.005
20.5	-25.193	-25.178	1.957	-0.015
22.0	-24.82	-24.821	1.909	0.001
25.0	-24.412	-24.39	1.981	-0.022
26.0	-24.088	-24.088	1.988	0
28.0	-22.908	-22.888	1.816	-0.02
30.0	-20.922	-20.933	1.349	0.011
33.0	-19.76	-19.774	1.203	0.014
35.0	-20.819	-20.804	1.569	-0.015
38.0	-21.265	-21.253	1.648	-0.012
42.0	-19.33	-19.31	1.484	-0.02
45.0	-14.505	-14.508	0.885	0.003
47.0	-10.018	-10.032	0.501	0.014
50.0	-5.558	-5.571	0.251	0.013
55.0	-2.927	-2.932	0.14	0.005
60.0	-0.499	-0.497	0	-0.002

References

Abdelrahman, E.M., Essa, K.S., 2015, Three least-squares minimization approaches to interpret gravity data due to dipping faults: Pure and Applied Geophysics, 172, 427-438.

Abdelrahman, E.M., Essa, K.S., Erabo-Ezz, 2013, A least-squares window curves method to interpret gravity data due to dipping faults: Journal of Geophysics and Engineering, 10, 025003.

Abdelrahman, E.M., El-Araby, H.M., El-Araby, T.M, Abo-Ezz, E.R., 2003, A least-squares derivatives analysis of gravity anomalies due to faulted thin slabs: Geophysics, 68, 535-543.

Abdelrahman, E.M., Bayoumi, A.I., El-Araby, H.M., 1989, Dip angle determination of fault planes from gravity data: Pure and Applied Geophysics, 130, 735-742.

Adema, G.W., Breckenridge, R.M., Sprenke, K.F., 2007, Gravity, morphology, and bedrock depth of the Rathdrum Prairie, Idaho, Technical Report, Idaho Geological Survey, University of Idaho.

Agarwal, B. N. P., Shalivahan, S., 2010, A FORTRAN code to implement the finite element method to compute regional and residual anomalies from the gravity data: Computers & Geosciences, 36, 848-852.

Agarwal, B. N. P., Sivagi, C., 1992, Separation of regional and residual anomalies by least squares orthogonal polynomial and relaxation techniques: a performance evaluation: Geophysical Prospecting, 40, 143-156.

Agarwal, B. P., 1995, Hydrocarbon prospects of the Pranhita - Godavari graben, India: Proceedings of Petrotech, 95, 115-121.

Albora, A.M., Ucan, O.N., Ozmen, A., 2001, Residual separation of magnetic fields using a cellular neural net-work approach, Pure and Applied Geophysics, 158, 1797-1818.

Al-Garni, M.A., Srinivas, Y., and Sundararajan, N., 2010, Sundararajan transform - an application to geophysical data analysis: Arabian Journal of Geosciences, 3, 27–32.

Ander, M.E., Huestis, S.P., 1987, Gravity ideal bodies: Geophysics, 52, 1265-1278.

Angell, H.T., Negrinia, R.M., 1997, Gravity Model of The Southern San Joaquin Basin Constrained by Density Data from a 24,000 Foot Well, Elk Hills, California, AAPG Pacific Section Meeting, Bakersfield, California.

Annecchione, M. A., Chouteau, M., Keating, P., 2001, Gravity interpretation of bedrock topography: the case of the Oak Ridges Moraine, southern Ontario, Canada: Journal of Applied Geophysics, 47, 63-81.

Artemjev, M.E., Kaban, M.K., Kucherinenko, V.A., Demyanov, G.V., Taranov, V.A., 1994, Subcrustal density inhomogeneities of Northern Eurasia as derived from the gravity data and isostatic models of the lithosphere: Tectonophysics, 240, 249–280.

Athy, L.F., 1930, Density, porosity and compaction of sedimentary rocks: Bulletin of the American Association of Petroleum Geology, 14, 1-24.

Aydemir, A., Ates, A., Bilim, F., Buyuksarac, A., Bektas, O., 2014, Evaluation of gravity and aeromagnetic anomalies for the deep structure and possibility of hydrocarbon potential of the region surrounding Lake Van, Eastern Anatolia, Turkey: Surveys in Geophysics, 35,431–448

Aydogan, D., 2011, Extraction of lineaments from gravity anomaly maps using the gradient calculation: Application to Central Anatolia: Earth Planets Space, 63, 903-913.

Azeglio, E.A., Gimenez, M.E., Introcaso, A., 2010, Interpretation of Las Salinas sedimentary basin - Argentina, based on integration of geological and geophysical data: Geofísica Internacional, 49, 225-244.

Baan, M.V.D., Jutten, C., 2000, Neural Networks in geophysics: Geophysics, 65, 1032-1047.

Bachman, R.T., Hamilton, E.L., 1976, Density, Porosity, and Grain Density of Samples From Deep Sea Drilling Project Site 222 (LEG 23) in the Arabian Sea: Journal of Sedimentary Petrology, 46, 654-658.

Backus, G.E., Gilbert, F.J., 1967, Numerical applications of a formalism for geophysical inverse problems: Geophysical Journal of the Royal Astronomical Society, 13, 247-276.

Backus, G.E., Gilbert, F.J., 1968, The resolving power of gross earth data: Geophysical Journal of the Royal Astronomical Society, 16, 169-205.

Backus, G.E., Gilbert, F.J., 1970, Uniqueness in the inversion of inaccurate gross earth data: Philosophical Transactions of the Royal of London, A226, 123-192.

Barbosa, V.C., Silva, J.B.C., Medeiros, W., 1997, Gravity inversion of basement relief using approximate equality constraints on depths: Geophysics, 62, 1745-1757.

Barbosa, V.C., Silva, J.B.C., Medeiros, W., 1999, Gravity inversion of a discontinuous relief stabilized by weighted smoothness constraints on depth: Geophysics, 64, 1429-1437.

Becking, L.G.M.B., Moore, D., 1959, Density Distribution in Sediments: Journal of Sedimentary Petrology, 29, 47-55.

Beiki, M., 2010, Analytic signals of gravity gradient tensor and their application to estimate source location: Geophysics, 75, I59–I74.

Beiki, M., Pedersen, L., 2010, Eigenvector analysis of gravity gradient tensor to locate geologic bodies: Geophysics, 75, I37–I49.

Beltrao, J.F., Silva, J.B.C., Costa, J.C., 1991, Robust polynomial fitting method for regional gravity estimation: Geophysics, 56, 80-89.

Bhaskara Rao, V., Venkateswarulu, P.D., 1974, A simple method of interpreting gravity anomalies over sedimentary basins: Geophysical Research Bulletin, 12, 177-182

Bhimasankaram, V.L.S., Nagendra, R., Seshagiri Rao, S.V., 1977, Interpretation of gravity anomalies due to finite inclined dikes using Fourier Transformation: Geophysics, 42, 51-59.

Blakely, R.J., 1995, Potential Theory in Gravity and Magnetic Applications: Cambridge University Press, 464 pp.

Bohidar, R.N., Sullivan, J.P., Hermance, J.F., 2001, Delineating depth to bedrock beneath shallow unconfined aquifers: a gravity transect across the Palmer river basin, Groundwater, 39, 729–736.

Boschetti, F., Dentith, M., List, R., 1997, Inversion of potential field data by genetic algorithms: Geophysical Prospecting, 45, 461-478.

Boschetti, F., Therond, V., Hornby, P., 2004, Feature removal and isolation in potential field data: Geophysical Journal International, 159, 833-841.

Bott, M. H. P., 1960, The use of rapid digital computing methods for direct gravity interpretation of sedimentary basins: Geophysical Journal of Royal Astronomical Society, 3, 63-67.

Brady, R., Wernicke, B., Fryxell, J., 2000, Kinematic evolution of a large-offset continental normal fault system, South Virgin Mountains, Nevada: Geological Society of America Bulletin, 112, 1375-1397.

Brown, M.P., POULTON, M.M., 1996, Location of buried objects for environmental site investigations using Neural Networks: Journal of Environmental & Engineering Geophysics, 1, 179-188.

Brun, J.P., Choukroune, P., 1983, Normal faulting, blocktilting, and décollement in a stretched crust: Tectonics, 2, 345-356.

Byerly, P.E., 1965, Convolution filtering of gravity and magnetic maps: Geophysics, 30, 281-283.

Cai, H., Zhdanov, M., 2015, Application of Cauchy-type integrals in developing effective methods for depth-to-basement inversion of gravity and gravity gradiometry data: Geophysics, 80, G81–G94.

Castagna, J.P., Batzle, M.L., Kan, T.K., 1993, Rock physics—the link between rock properties and AVO response. In Offset-Dependent Reflectivity—Theory and Practice of AVO Analysis, ed. P. Castagna and M.M. Backus, No. 8, 124–157. Tulsa, Oklahoma: Investigations in Geophysics series, Society of Exploration Geophysicists.

Chacko, S., Battacharya, B., 1980, A method for analyzing gravity anomalies due to a geologic contact by Fourier transform: Geoexploration, 18, 43-50.

Chai, Y., Hinze, W.J., 1988, Gravity inversion of an interface above which the density contrast varies exponentially with depth: Geophysics, 53, 837-845.

Chaika, C., Williams, L.A., 2001, Density and mineralogy variations as a function of porosity in Miocene Monterey Formation oil and gas reservoirs in California, Bulletin of the American Association of Petroleum Geologists, 85, 149-167.

Chakraborty, C., Mandal, N., Ghosh, S.K., 2003, Kinematics of the Gondwana basins of peninsular India: Tectonophysics, 377, 299–324.

Chakravarthi, V., Pramod Kumar, M., Ramamma, B., Rajeswara Sastry, S., 2015a, Gravity anomaly interpretation of 2D listric fault morphologies using exponential density contrast model: A space domain technique: Arabian Journal of Geosciences (under review).

Chakravarthi, V., Pramod Kumar, M., 2015b, Estimation of multiple density-depth parameters from gravity inversion: Application to detached hanging wall systems of strike limited listric fault morphologies: Geophysica Internacional (Springer), 54, 49-65.

Chakravarthi, V., Pramod Kumar, M., Ramamma, B., Rajeswara Sastry, S., 2015c, Automatic gravity modeling of sedimentary basins by means of polygonal source geometry and exponential density contrast variation: Two space domain based algorithms: Journal of Applied Geophysics (in press).

Chakravarthi, V., Rajeswara Sastry, S., Pramod Kumar, M., 2014, A method and a GUI based JAVA code for interactive gravity modeling of strike limited listric fault sources with arbitrary density-depth variations: Journal of the Geological Society of India, 83, 577-585.

Chakravarthi. V., Rajeswara Sastry, S., Ramamma, B., 2013a, MODTOHAFSD – A GUI based JAVA code for gravity analysis of strike limited sedimentary basins by means of growing bodies with exponential density contrast – depth variation: A space domain approach: Computers & Geosciences, 56, 131-141.

Chakravarthi. V., Ramamma, B., 2013b, Gravity anomaly modelling of multiple geological sources having differing strike lengths and arbitrary density contrast variations: Near Surface Geophysics, 11, 363-370.

Chakravarthi, V., Ramamma, B., Venkat Reddy, T., 2013c, Gravity Anomaly Modeling of Sedimentary Basins by Means of Multiple Structures and Exponential Density Contrast-depth Variations: A Space Domain Approach: Journal of the Geological Society of India, 82, 561-569.

Chakravarthi, V., 2011a, Geodesy, Physical: Encyclopedia of Solid Earth Geophysics (Springer), 331-335.

Chakravarthi, V., 2011b, Automatic gravity optimization of 2.5D strike listric fault sources with analytically defined fault planes and depth-dependent density: Geophysics, 76, I21-I31.

Chakravarthi, V., 2010a, LSTRKFALTG — A forward modeling program to compute theoretical gravity anomalies of strike limited listric fault structures with prescribed vertical variation in density: Computers & Geosciences, 36, 675–679.

Chakravarthi, V., 2010b, Gravity anomalies of strike limited listric fault sources with analytically defined fault planes and arbitrary density contrast variations with depth: Near Surface Geophysics, 8, 279-286.

Chakravarthi, V., 2009, Automatic gravity inversion for simultaneous estimation of model parameters and regional gravity background: an application to 2D pull-apart basins: Current Science, 96, 1349-1360.

Chakravarthi. V., 2008, Gravity inversion of 2.5D faulted beds using depth dependent density, Current Science, 95, 1618-1622.

Chakravarthi, V., Sundararajan, N., 2007, 3D gravity inversion of basement relief – A depth dependent density approach: Geophysics, 72, I23- I32.

Chakravarthi, V., Sundararajan, N., 2006, Discussion on "The gravitational attraction of a right rectangular prism with density varying with depth following a cubic polynomial" by García-Abdeslem, J (November-December 2005, Geophysics, j39-j42), Geophysics, 71, X17–X19.

Chakravarthi, V., Sundararajan, N., 2004, Ridge regression algorithm for gravity inversion of fault structures with variable density: Geophysics, 69, 1394-1404.

Chakravarthi, V., 2003, Digitally implemented method for automatic optimization of gravity fields obtained from three-dimensional density interfaces using depth dependent density: US Patent 6,615,139.

Chakravarthi, V., Singh S.B., AshokBabu, G., 2001, INVER2DBASE - A program to compute basement depths of density interfaces above which the density contrast varies with depth: Computers & Geosciences, 27, 1127–1133.

Chapin, D. A., 1996, A deterministic approach toward isostatic gravity residuals; a case study from South America: Geophysics, 61, 1022 -1033.

Chappell, A., Kusznir, N., 2008, An algorithm to calculate the gravity anomaly of sedimentary basins with exponential density-depth relationships: Geophysical Prospecting, 56, 249–258.

Chávez, R.E., Lazaro-Mancilla, O., Campos-Enríquez, J.O., Flores-Márquez, E.L., 1999, Basement topography of the Mexicali Valley from spectral and ideal body analysis of gravity data: Journal of South American Earth Sciences, 12, 579–587.

Christiansen, A.F., 1983, An example of a major syndepositional listric fault. In: Bally AW (ed) Seismic Expression of Structural Styles: American Association of Petroleum Geologists Studies in Geology, 15, 36-40.

Collins, S.J., Dodds, A.R., Johnson, B.D., 1974, Gravity profile interpretation using Fourier transform: Geophysics, 39, 862-866.

Constenius, K.N., 1996, Late Paleogene extensional collapse of the Cordil- leran foreland fold and thrust belt: Geological Society of America Bulletin, 108, 20- 30.

Cordell, L., 1973, Gravity anomalies using an exponential density-depth function-San Jacinto Graben, California: Geophysics, 38, 684-690.

Corner, B., Wilsher, W.A., 1989, Structure of the Witwatersrand basin derived from interpretation of the aeromagnetic and gravity data, in Proceedings of exploration '87, Third decennial international conference on geophysical and geochemical exploration for minerals and groundwater, Ontario: Geological Survey, 3, 532–546.

Cowie, P.A., Karner, G.D., 1990, Gravity effect of sediment compaction: examples from the North Sea and Rhine Graben: Earth and Planetary science Letters, 99, 141-153.

Dallmus, K.F., 1958, Mechanics of basin evolution and its relation to the habitat of oil in the basin: In Habitat of Oil, L.G. Weeks, No. 36, 2071–2174. Tulsa, Oklahoma: AAPG Memoir, American Association of Petroleum Geologists.

De Vicente, J., Lanchares, J., Hermida, R., 2003, Placement by thermodynamic simulated annealing: Physics Letters, A317, 415-423.

Dickinson, G., 1953, Geological Aspects of Abnormal Reservoir Pressures in Gulf Coast Louisiana: Bulletin-American Association of Petroleum Geologists, 37, 410-432.

Dimitropoulos, K., Donato, J.A., 1981, The inner moray firth central ridge, a geophysical interpretation: Scottish Journal of Geology, 17, 27–38.

Donato, J.A., Tully, M.C., 1981, A regional interpretation of North Sea gravity data, in Petroleum Geology of the Continental Shelf of North-West Europe: Eds. L.V.Illing and G.D.Hobson, 65–75.

Dorigo, M., Blum, C., 2005, Ant colony optimization theory: a survey: Theoretical Computational Science, 344, 243-278.

D'Urso, M.G., 2014a, Gravity effects of polyhedral bodies with linearly varying density: Celestial Mechanics and Dynamical Astronomy, http://library.seg.org/doi/abs/10.1190/1.1440973

D'Urso, M.G., 2014b, Analytical computation of gravity effects for polyhedral bodies: Journal of Geodesy, 88, 13–29.

Eaton, B.A., 1969, Fracture gradient Prediction and Its Application in Oilfield Operations: Journal of Petroleum Technology and Alternative Fuels, 21,1353–1360.

Eberhart, R.C., Kennedy, J., 1995, A new optimizer using particle swarm theory: Proceedings of sixth International symposium on micro machine and human science, IEEE service centre, Piscataway, Nagoya, Japan, 39-43.

Emeleus, C.H., 1981, A thrust fault at Welshman's Rock, Isle of Rhum, Scottish: Journal of Geology, 17, 1-6.

Essa, K.S., 2013, Gravity interpretation of dipping faults using the variance analysis method: Journal of Geophysics and Engineering, 10: 015003.

Fairhead, J.D., Bennett, K.J., Gordon, R.H., Huang, D., 1994, Euler: Beyond the 'BlackBox', 64th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 422–424.

Fett, J.D., 1968, Geophysical investigation of the San Jacinto Valley, Riverside County, California: Unpublished thesis, University of California at Riverside, 87 pp.

Forsythe, G.E., 1957, Generation and use of orthogonal polynomials for data filtering with a digital computer: Journal Society Indian Applied Mathematics, 5, 75-88.

Foucher, J.P., LePichon,X., Sibuet, J.C., 1982, The ocean-continent transition in the uniform stretching model: role of partial melting in the mantle: Philosophical Transactions of the Royal Society A, 305, 1489, 27–41.

Gabalda, G., Nalpas, T., Bonvalot, S., 2005, The Base of the Atacama Gravels Formation (26°S, Northern Chile): first results from gravity data, 6th International Symposium on Andean Geodynamics (ISAG2005, Barcelona). IRD, Paris, 286–289 (Extended Abstracts).

Ganguly, S.S., Dimri, V.P., 2013, Interpretation of gravity data using eigenimage with Indian case study: A SVD approach: Journal of Applied Geophysics, 95, 23-35.

Gans, P.B., Miller, E.L., McCarthy, J., Ouldcott, M.L., 1985, Tertiary extensional faulting and evolving ductile-brittle transition zones in the northern Snake Range and vicinity: New insights from seismic data, Geology, 13, 189-193.

García-Abdeslem, J., 2003, 2D modeling and inversion of gravity data using density contrast varying with depth and source-basement geometry described by the Fourier series: Geophysics, 68, 1909-1916.

Gealy, E., 1969, Saturated bulk density, grain density, and porosity of sediment cores from the western equatorial Pacific, Glomar Challenger, Initial Report of the Deep Sea Drilling Project, VII, part 2.

Geldart, L.P., Gill, D.E., Sharma, B., 1966, Gravity anomalies of two dimensional faults: Geophysics, 31, 372-397.

Gibbs, A.D., 1983, Balanced cross-section construction from seismic sections in areas of extensional tectonics: Journal Structural Geology, 5, 153- 160.

Goussev, S., Charters, R., Peirce, J., 2006, Mackenzie Delta: A case of one residual gravity anomaly and 16 dry exploration wells: CSPG/ CSEG/ CWLS Joint Conference, Calgary, Canada, Expanded Abstracts, 435-439.

Grandis, H., Dahrin, D., 2014, Constrained two-dimensional inversion of gravity data: Journal of mathematical and fundamental sciences, 46, 1-13.

Granser, H., 1987, Three-dimensional interpretation of gravity data from sedimentary basins using an exponential density-depth function: Geophysical Prospecting, 35, 1030-1041.

Grant, F.S., 1957, A problem in the analysis of geophysical data: Geophysics, 22, 309-344.

Gu, X., Tenzer, R., Gladkikh, V., 2014, Empirical models of the ocean-sediment and marine sediment-bedrock density contrasts: Geosciences Journal (in press), DOI 10.1007/s12303-014-0015-9.

Guang, Z., Hou, J., Huang, L., Yao, C., 1998, Inversion of gravity and magnetic anomalies using pseudo-BP neural network method and its application: Chinese Journal of Geophysics, 41, 139-149.

Gupta, D.K., Gupta, J.P., Arora, Y., Singh, U.K., 2011, Recursive Ant Colony Global Optimization: a new technique for the inversion of geophysical data: American Geophysical Union, Fall Meeting, abstract #H13B-1196.

Gupta, O.P., Pokhriyal. S.K., 1990, New formula for determining the dip angle of a fault from gravity data: SEG Technical Program Expanded Abstracts 9646–9.

Gupta, O.P., 1983, A least-squares approach to depth determination from gravity data: Geophysics, 48 (3), 357-360

Guspí, F., 1990, General 2D inversion with density contrast varying with depth: Geoexploration, 26, 253-265.

Hamayun, Prutkin, I., Tenzer, 2009, The optimum expression for the gravitational potential of polyhedral bodies having a linearly varying density distribution: Journal of Geodesy, **83,** 1163-1170

Hamilton, E.L., Menard, H.W., 1956, Density and porosity of seafloor sediments off San Diego, California: Bulletin of the American Association of Petroleum Geologists, 40, 754-761.

Hamilton, E.L., 1976, Variations of density and porosity with depth in deep-sea sediments: Journal of Sedimentary Petrology, 46, 1-2, 280–300.

Hansen, R.O., 1999, An analytical expression for the gravity field of a polyhedral body with linearly varying density: Geophysics, 64, 75–77.

Helmberger, D.V., Engen, G., Scott, P., 1979, A note on velocity, density, and attenuation models for marine sediments determined from multi-bounce phases: Journal of Geophysical Research, 84, 667–671.

Hinze, W.J., Aiken, C., Brozena, J., Coakley, B., Dater, D., Flanagan, G., Forsberg, R., Hildenbrand, T., Keller, G. R., Kellogg, J., Kucks,R., Li, X., Mainville, A., Morin, R., Pilkington, M., Plouff, D., Ravat, D., Roman, D., Urrutia-Fucugauchi, J., Veronneau, M., Webring,M., Winester, D., 2005, New standards for reducing gravity data: The North American gravity database: Geophysics, 70, J25-J32.

Hood, P., 1965, Gradient measurements in aeromagnetic surveying, Geophysics, 30, 891–802.

Holliger, K., Klemperer, S.L., 1989, A comparison of the Moho interpreted from gravity data and from deep seismic reflection data in the northern North Sea, Geophysical Journal International, 97, 247–258.

Hu, C., Fan, M., Wu, Y., Pei, Y., 2011, Euler deconvolution method of gravity anomaly based on wavelet analysis: International Conference on Multimedia Technology (ICMT 2011), Hangzhou, China, 305-308.

Huang, D., Gubbins, D., Clark, R.A., Whaler, K.A., 1995, Combined study of Euler's homogeneity equation for gravity and magnetic field, 57th Conference & Technical Exhibition: European Association of Exploration Geophysicists, Extended Abstracts, 144.

Huestis, S.P., Parker, R.L., 1977, Bounding the thickness of the oceanic magnetized layer: Journal of Geophysical Research, 82, 5293–5303.

Inoue, N., Kitada,N., Takemura,K, 2012, Blind fault Configuration in Osaka, Japan based on 2D gravity inversion, 15 WCEE conference (http://www .iitk.ac.in/nicee/wcee/article/WCEE2012_2615.pdf).

Jackson, J.A., 1987, Active normal faulting and crustal extension. Geol. Soc. London Spec. Publ., 28, 3–17.

Jackson, J.A. McKenzie, D., 1983, The geometrical evolution of normal fault systems: Jour. Struct. Geol., 5, 471-482.

Jacobsen, B.H., 1987, A case for upward continuation as a standard separation filter for potential field maps: Geophysics, 52, 1138-1148.

Jacoby, W., Smilde, P.L., 2009, Gravity interpretation—fundamentals and application of gravity inversion and geological interpretation: Springer, Berlin.

Janecke, S.U., Vandenburg, C.J. Blankenau, J.J., 1998, Geometry, mechanisms and significance of extensional folds from examples in the Rocky Mountain Basin and Range province, U.S.A., Jour. Struct. Geol., 20, 841-856.

Jingxin, Q., Ziqi, G., Jianying, L., Yanchao, Q., 2013, Research in Modified Simulated Annealing Interface Inversion Method: Near Surface Geophysics Asia Pacific Conference, Beijing, China, 657-659.

Kadirov, F.A., 2000, Application of the Hartley transform for interpretation of gravity anomalies in the Shamakhy–Gobustan and Absheron oil-and gas-bearing regions, Azerbaijan: Journal of Applied Geophysics, 45, 49-61.

Kaila, K.L., Murthy, P.R.K., Rao, V.K., Venkateswarlu, N., 1990, Deep seismic sounding in the Godavari Graben and Godavari (Coastal) Basin, India: Tectonophysics, 173, 307–317.

Kearey, P., Brooks, M., Healy I., 2002, An introduction to Geophysical exploration, third ed. Blackwell Science.

Kennedy, J., Eberhart, R., 1995, Particle swarm optimization, IEEE International Conference on Neural Networks, 4, 1942–1948.

Klingele, E.E., Marson, I., Kahle, H.G., 1991, Automatic interpretation of gravity gradiometric data in two dimensions: Vertical gradients, Geophysical Prospecting, 39, 407–434.

Krahenbuhl, R.A., Li, Y.G., 2006, Inversion of gravity data using a binary formulation: Geophysical Journal International, 167, 543–556.

Krimmel, R.M., 1970, Gravimetric ice thickness determination, South Cascade Glacier, Washington: Northwest Science, 44, 147–153.

Kwok, Y.K., 1991, Contour integrals for gravity computation of horizontal $2^{1/2}$-D bodies with variable density. Applied mathematical modeling 15, 98-103.

LaFehr, T.R., Nabighian, M.N., 2012,Fundamentals of gravity exploration, Society of Exploration Geophysicists, USA.

Lavin, P.M., Devane, J.F., 1970, Direct design of two-dimensional digital wave number filters: Geophysics, 35, 1073-1078.

Leão, J.W.D., Menezes, P.T.L., Beltrão, J.F., Silva, J.B.C., 1996, Gravity inversion of basement relief constrained by the knowledge of depth at isolated points: Geophysics, 61, 1702–1714.

Leite, E.P., Filho, C.R., de. S., 2009, Artificial neural networks applied to mineral potential mapping for copper-gold mineralizations in the Carajás Mineral Province, Brazil: Geophysical Prospecting 57, 1049-1065.

Levy, A.V., Montalvo, A., 1985, The Tunneling algorithm for the global minimization of functions: SIAM Journal on Scientific and Statistical Computing, 6, 15–29.

Li, Y., Oldenburg, D.W., 1998, 3-D gravity inversion of gravity data: Geophysics, 63, 109-119.

Mallick, K., Sharma, K.K., 1999, A finite element method for computation of the regional gravity anomaly: Geophysics, 64, 461-469.

Manger, G.E., 1963, Porosity and bilk density of sedimentary rocks, U.S. Geological Survey Bulletin, 1144-E, E1-E56.

Mantlík, F., Matias, M., Lourenço ,J., Grangeia, C., Tareco, H., 2009, The use of gravity methods in the internal characterization of landfills – a case study: Journal of Geophysics and Engineering, 6,357–364.

Mareschal, J.C., 1985, Inversion of potential field data in Fourier transform domain: Geophysics, 50, 685-691.

Marlet, G., Sailhac, P., Moreau, F., Diament, M., 2001, Characterization of geological boundaries using 1-D wavelet transform on gravity data; theory and application to the Himalayas: Geophysics, 66, 1116-1129.

Marson, I., Klingele, E.E., 1993, Advantages of using the vertical gradient of gravity for 3-D interpretation: Geophysics, 58, 1588–1595.

Marquardt, D.W., 1963, An algorithm for least squares estimation of nonlinear parameters: Journal Society Indian Applied Mathematics, 11, 431-441.

Marquardt, D.W., 1970, Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation: Technometrics, 12, 591-612.

Martín Atienza, B., García-Abdeslem, J., 1999, 2-D gravity modeling with analytically defined geometry and quadratic polynomial density functions: Geophysics, 64, 1730–1734.

Martinez, P., Gimenez, M., Folguera, A., Klinger, F.L., 2014, Integrated seismic and gravimetric model of Jocolí Basin, Argentina: Interpretation, 2, T57–T68.

Maxant, J., 1980, Variation of density with rock type, depth, and formation in the Western Canada basin from density logs: Geophysics, 45, 1061-1076.

McCulloh, T.H. 1967, Mass properties of sedimentary rocks and gravimetric effects of petroleum and natural-gas reservoirs. USGS Professional Paper 528-A, Department of the Interior, United States Geological Survey, Washington, DC.

McGrath, P.H., 1991, Dip and depth extent of density boundaries using horizontal derivatives of upward-continued gravity data: Geophysics, 56, 1533-1542.

McKenzie, D., Jackson, J., 2012, Tsunami earthquake generation by the release of gravitational potential energy: Earth and Planetary Science Letters, 345-348, 1-8.

McNeill, L.C., Piper, K.A., Goldfinger, C., Kulm, L.D., Yeats, R.S., 1997, Listric normal faulting on the Cascadia continental margin: Journal of Geophysical Research 102: 123-12,138.

Mendonca, C.A., 2004, Inversion of gravity-field inclination to map the basement telief of sedimentary basins: Geophysics, 69, 1240-1251.

Merriam, D.F., Cocke, N.C., 1967, Colloquium on trend analysis: Kansas State Geological survey Comput. Contr., 12, P.62.

Mickus, K.L., Aiken, C.L.V., Kennedy, W.D., 1991, Regional-residual gravity anomaly separation using minimum curvature technique: Geophysics, 56, 279-283.

Mickus, K.L., Peeples, W.J., 1992, Inversion of gravity and magnetic data for the lower surface of a two and one-half dimensional sedimentary basin: Geophysical Prospecting, 40, 171-193.

Mishra, D.C., Gupta, S.B., Rao, M.B.S.V., Venkatarayudu, M., Laxman, G., 1987, Godavaribasin - a geophysical study: Journal of the Geological Society of India, 30, 469–476.

Mohan, N.L., 1978, Interpretation techniques in geophysical exploration using Fourier Transforms: Ph. D thesis, Osmania University, Hyderabad, India (unpublished).

Mohan, C.R., Sastry, R.G.S., Moharir, P.S., 1986, Inversion of gravity anomalies using Tunneling algorithm: In twelfth annual convention and seminar on Exploration Geophysics, A7-A8.

Moharir, P.S., 1990, Inversion of potential field data, Journal of Earth System Science, 99, 473-514.

Moritz, H., 1980, Advanced Physical Geodesy: Karlsruhe:Wichmann.

Mosegaard, K., Tarantola, A., 1995, Monte Carlo sampling of solutions to inverse problems: Journal of Geophysical Research, 100, 12431-12447.

Mundim, K.C., Lemaire, T.J., Bassrei, A., 1998, Optimization of non-linear gravity models through generalized simulated annealing, Physica A, 252, 405–416.

Murty, B.V.R., Parthasarathy, E.V.R., 1988, On the evolution of the Godavari Gondwana Graben based on LANDSAT Imagery interpretation, Journal of the Geological Society of India, 32, 417–425.

Murthy, I.V.R., 1998, Gravity and Magnetic Interpretation in Exploration Geophysics: Memoir 40, Geological Society of India.

Murthy, I.V.R., Krishnamacharyulu, S.K.G., 1990, Automatic inversion of gravity anomalies of faults: Computers & Geosciences, 16, 539-548.

Murthy, I.V.R., Rao, P.R., Rao, S.J., 1990, The density difference and generalized programs for two and three-dimensional gravity modeling: Computers & Geosciences,16, 277-287.

Murthy, I.V.R., Rao, S.J., 1989, A Fortran 77 program for inverting gravity anomalies of two-dimensional basement structures: Computers & Geosciences, 15, 1149-1156.

Murthy, I.V.R., Krishna, P.R., Rao, S.J., 1988, A generalized computer program for two-dimensional gravity modeling of bodies with a flat top or a flat bottom or undulating over a mean depth: Journal of Association of Exploration Geophysicists, 9, 93-103.

Murthy, I.V.R., Rao, D.B., 1980, Interpretation of gravity anomalies over faults and dykes by Fourier transforms, employing end correction: Geophysical Research Bulletin, 18, 95-110.

Murthy, I.V.R., Rao, D.B., 1979, Gravity anomalies of two-dimensional bodies of irregular cross-section with density contrast varying with depth: Geophysics, 44, 1525-1530.

Mushayandebvu, M.F., Driel, P.V., Reid, A.B., Fairhead, J.D., 2001, Magnetic source parameters of two-dimensional structures using extended Euler deconvolution: Geophysics, 66, 814–823.

Nagihara, S., Hall, S.A., 2001, Three-dimensional gravity inversion using simulated annealing, constraints on the diapiric roots of allochthonous salt structures: Geophysics, 66, 1438-1449.

Naidu, P.S., 1966, Extraction of a potential field signal from a background of random noise by Strakhov's method: Journal of Geophysical Research, 71, 5987-5995.

Naidu, P.S., 1967, Two-dimensional Strakhov's filter for extraction of a potential field signal: Geophysical Prospecting, 15, 135-150.

Naudy, H., Dreyer, H., 1968, Essai de filtrage non-lineaire appliqué aux profiles aeromagnetiques: Geophysical Prospecting, 16, 171-178.

Odegard, M.E., 2011, A sediment thickness map of South America using automated inversion of magnetic and gravity data for depth to basement: 12th International Congress of the Brazilian Geophysical Society and EXPOGEF, SEG and Brazilian Geophysical Society, 581–586.

Odegard, M.E., Berg, J.W., 1965, Gravity interpretation using the Fourier integral: Geophysics, 30(3), 424-438.

Okabe, M., 1979, Analytical expressions for gravity anomalies due to homogeneous polyhedral bodies and translations into magnetic anomalies: Geophysics, 44, 730-741.

Oliva, S.E. Ravazzoli, C.L., 1997, Complex polynomials for the computation of 2D gravity anomalies: Geophysical Prospecting 45, 809–818.

Oruç, B., 2014, Structural interpretation of south part of western Anatolian using analytic signal of the second order gravity gradients and discrete wavelet transform analysis: Journal of Applied Geophysics (In Press, Accepted Manuscript), Available online 24 January 2014.

Osman, O., Albora, A.M., Ucan, O. N., 2006, A new approach for residual gravity anomaly profile interpretations: Forced Neural Network (FNN): Annals of Geophysics, 49, 1201-1208.

Osman, O., Albora, A.M., Ucan, O.N., 2007, Forward modeling with forced neural networks for gravity anomaly profile: Mathematical Geology, 39, 593–605.

Pacino, M.C., Introcaso, A., 1988, Regional anomaly determination using the upwards – continuation method: Bollettino di Geofisica Teorica e Applicata,, XXIX (114), 113- 122.

Pal, P.C., 1981, Gravity profile interpretation using Fourier transform: Geoexploration, 19, 167-177.

Pallero, J.L.G., Fernández-Martínez, J.L., Bonvalot, S., Fudym, O., 2015, Gravity inversion and uncertainty assessment of basement relief via Particle Swarm Optimization: Pure and Applied Geophysics, 116, 180–191.

Pan J.J., 1989. Gravity anomalies of irregularly shaped two-dimensional bodies with constant horizontal density gradient: Geophysics 54, 528-530.

Parker, R.L., 1974, Best bounds on density and depth from gravity data: Geophysics, 39, 644-649.

Parker, R.L., 1975, The theory of ideal bodies for gravity interpretation: The Geophysical journal of the Royal Astronomical Society**,** 42, 315–334.

Paul, M.K., Datta S., Banerjee, B., 1966, Direct interpretation of two dimensional structural fault from gravity data: Geophysics, 31, 940-948.

Pawlowski, R.S., 1994, Green's equivalent-layer concept in gravity band-pass filter design: Geophysics, 59, 69-76.

Pawlowski, R.S., Hansen, R.O., 1990, Gravity anomaly separation by Wiener filtering: Geophysics, 55, 539–548.

Peirce, J. W., Lipkov, L., 1988, Structural interpretation of the Rukwa Rift, Tanzania: Geophysics, 53, 824-836.

Pilkington M., Crossley D.J., 1986, Determination of crustal interface topography from potential fields: Geophysics 51, 1277–1284.

Prozorovich, E.A., 1960, Density and Porosity of Clayey Rocks Under Differing Conditions of Moisture, Petroleum Geology: A Digest of Russian Literature, 4, 2B, 98-104.

Qureshy, M.N., KrishnaBrahmam, N., Garde, S.C., BMathur, .K., 1968, Gravity anomalies and the Godavaririft,India: Geological Society of America, 79, 1221–1230.

Ramakrishna, T.S., Chayanulu, A.Y.S.R., 1988, A geophysical appraisal of the Purana basins of India: Journal of the Geological Society of India, 32, 48–60.

Rao, B.S.R., Murthy, I.V.R., 1978, Gravity and Magnetic methods of prospecting: Arnold-Heinemann publishers (India) Pvt. Ltd., 390 pp.

Rao, D.B., 2013, Gravity anomalies of two-dimensional bodies: Journal of Indian Geophysical Union, 17, 129-137.

Rao, D.B., 1990, Analysis of gravity anomalies of sedimentary basins by an asymmetrical trapezoidal model with quadratic density function: Geophysics, 55, 226-231.

Rao, D.B., 1986, Modeling of sedimentary basins from gravity anomalies with variable density contrast: Geophysical Journal of the Royal Astronomical Society, 84, 207-212.

Rao, D.B., 1985 Analysis of gravity anomalies over an inclined fault with quadratic density function: Pure and Applied Geophysics, 123, 250-260.

Rao, D.B., Prakash, M.J., 1990, Interpretation of gravity anomalies over an inclined fault of finite strike length with quadratic density function: Australian Journal of Exploration Geophysics, 21, 169-173.

Rao, D.B., Prakash, M.J., Babu, N.R., 1993, Gravity interpretation using Fourier transforms and simple geometrical models with exponential density contrast: Geophysics, 58, 1074-1083.

Rao, P.R., Murthy, I.V.R., 1989. Two fortran77 function subprograms to calculate gravity anomalies of bodies of finite and infinite strike length with the density contrast differing with depth: Computers and Geosciences 15, 1265–1277.

Rao, M.M.M., Murthy, T.V.R., Murthy, K.S.R. Vasudeva, R.Y., 2003, Application of natural generalized inversetechnique in reconstruction of gravity anomalies due to a fault: Indian Jour. Pure Appld. Mathematics, 34, 31-47.

Reid, A.B., 1997, Euler Deconvolution, Past, Present and Future: A Review, Proceedings of Exploration 97: Fourth Decennial International Conference on Mineral Exploration, 861–864.

Reid, A.B., Allsop, J.M., Granser, H., Millett, A.J., Somerton, I.W., 1990, Magnetic interpretation in three dimensions using Euler deconvolution: Geophysics, 55, 80–91.

Reamer, S.K., Ferguson J.F., 1989, Regularized two-dimensional Fourier gravity inversion method with application to the Silent Canyon caldera, Nevada: Geophysics, 54, 486-496.

Rieke, H.H., Chilingarian, G.V., 1974, Compaction of Argillaceous Sediments. Amsterdam, The Netherlands: Elsevier Scientific Publishing Company.

Roy, A., 1962, Ambiguity in geophysical interpretation: Geophysics, 27, 90-99.

Roy, L., Shaw, R., Agarwal, B.N.P., 2002, Inversion of gravity anomalies over sedimentary basins; applications of genetic algorithm and simulated annealing: SEG Annual Meeting Expanded Technical Program Abstracts with Biographies, 72, 747-750.

Sax, R.L., 1966, Application of filter theory and information theory to the interpretation of gravity measurements: Geophysics, 31, 570-575.

Shalivahan, S., Agarwal, B.N.P., 2010, Inversion of the amplitude of the analytic signal of the magnetic anomaly by particle swarm optimization technique: Geophysical Journal International, 182, 652-662.

Sharma, B., Geldart, L.P., 1968, Analysis of gravity anomalies using Fourier transforms: Geophysical Prospecting, 16, 77-93.

Shaw, R.K., Agarwal, B.N.P., 1990, The application of Walsh transforms to interpret gravity anomalies due to some simple geometrically shaped causative sources: A feasibility study: Geophysics, 55, 843-850.

Sigl, R., 1985, Introduction to Potential Theory: Cambridge, Abacus Press.

Silva, J.B.C., Oliveira, A.S., Barbosa, V.C.F., 2010, Gravity inversion of 2D basemen trelief using entropic regularization: Geophysics,75, I29–I35.

Silva, J.B.C., Medeiros, W.E., Barbosa, V.C.F., 2001, Potential-field inversion: Choosing the appropriate technique to solve a geologic problem: Geophysics, 66, 511–520.

Silva Dias, F.J.S., Barbosa, V.C.F., Silva, J.B.C., 2007, 2D gravity inversion of a complex interface in the presence of interfering sources: Geophysics, 72, I13–I22.

Singh, B., 2002, Simultaneous computation of gravity and magnetic anomalies resulting from a 2-D object: Geophysics, 67,801–806.

Smith, R.B,, Bruhn, R.L., 1984, Intraplate extensional tectonics of the Eastern Basin-Range: Inferences on structural style from seismic reflection data, regional tectonics, and thermal-mechanical models of brittle-ductile deformation: Journal of Geophysical Research, 89, 5733-5762.

Spector, A., Grant, F.S., 1970, Statistical models for interpreting aeromagnetic data: Geophysics, 35, 293-302.

Spitz, O.T., 1966, Generation of orthogonal polynomials for trend surfacing with a digital computer: Proceedings of symposium on operations research, In Mineral Ind., Pennsylvania State University, 3, 2-7.

Srivastava, S., Datta, D., Agarwal, B.N.P., Mehta, S., 2013, Applications of Ant Colony Optimization in determination of source parameters from total gradient of potential fields: Near Surface Geophysics (Early Online), DOI: 10.3997/1873-0604.2013054.

Stavrev, P., Reid, A., 2010, Euler deconvolution of gravity anomalies from thick contact/fault structures with extended negative structural index: Geophysics, 75, I51-I58.

Stern, T.A., 1978, Gravity survey of the Taylor Glacier, Victoria Land, Antarctica: Technical Report 8, Geology Department, Victoria University of Wellington.

Storer, D., 1959, Compaction of the argiuaceous sediments in the Padano Basin: In The Gasiferous Deposits of Western Europe, 2, 519–536. Roma, Italy.

Strakhov, V.N., 1964, The smoothing of observed strength of potential fields: Akad. Nauk. SSSR. Izv. Fizika Zemli, Part I, 897-904.

Strakhov, V.N., Lapina, M.I., 1967, A method of smoothing potential fields: Akad. Nauk. SSSR. Izv. Fizika Zemli, 40-57.

Sundararajan, N., Brahmam, G.R., 1998, Spectral analysis of gravity anomalies caused by slab-like structures: A Harley transform technique: Journal of Applied Geophysics, 39, 53-61.

Sundararajan, N., Srinivas, Y., Rao, T.L., 2000, Sundararajan Transform - a tool to interpret potential field anomalies: Exploration Geophysics, 31, 622 – 628.

Sundararajan, N., Mohan, N.L., Rao, S.V.S., 1983, Gravity interpretation of 2D fault structures using Hilbert transforms: Journal of Geophysics, 53, 34-41.

Talwani, M., Worzel, J., Ladisman, M., 1959, Rapid gravity computations for two dimensional bodies with application to the Mendocino submarine fracture zone: Journal of Geophysical Research, 64, 49 - 59.

Tankard, A.J., Welsink, H.J., 1987, Extensional tectonics and stratigraphy of Hibernia oil field, Grand Banks, Newfoundland: American Association of Petroleum Geologists Bulletin, 71, 1210-1232.

Tarantola, A., 2005, Inverse Problem Theory and Model Parameter Estimation: Society for Industrial and Applied Mathematics.

Telford, W.M., Geldert, L.P., Sheriff, R.E., 1990, Applied Geophysics, Cambridge University Press.

Tenzer, R., Gladkikh, V., 2014, Assessment of Density Variations of Marine Sediments with Ocean and Sediment Depths: The Scientific World Journal, Article ID 823296, http://dx.doi.org/10.1155/2014/823296.

Thanassoulas, C., Tselentis, G.A., Dimitriadis, K., 1987, Gravity inversion of a fault by Marquardt's method: Computers& Geosciences, 13, 399–404.

Thompson, D.T., 1982, EULDPH – A new technique for making computer assisted depth estimates from magnetic data: Geophysics, 47, 31–37.

Thorne, J.A., Watts, A.B., 1989, Quantitative analysis of North Sea subsidence: The American Association of Petroleum Geologists Bulletin, 73, 88–116.

Torizin, J., Jentzsch, G., Malischewsky, P., Kley, N., Ab, J., akanov, Kurskeev, A., 2009, Rating of seismicity and reconstruction of the fault geometries in northern Tien Shan within the project “Seismic Hazard Assessment for Almaty”:Journal of Geodynamics, 48, 269–278.

Toushmalani, R., 2013, Gravity inversion of a fault by Particle swarm optimization (PSO): Springer Plus, 2, 315.

Venteris, E., Miller, M., 1993, Gravitational profiles on the Taku Glacier System, Glaciological and Arctic Sciences Institute, University of Idaho, Open File Report.

Visweswara Rao, C., Chakravarthi, V., Raju, M.L., 1994, Forward Modelling: Gravity anomalies of two-dimensional bodies of arbitrary shape with hyperbolic and parabolic density functions: Computers and Geosciences, 20, 873-880.

Wang, T., Tucholke, J.L.B., Chen, Y.J., 2011, Crustal thickness anomalies in the North Atlantic Ocean basin from gravity analysis: Geochemistry, Geophysics, Geosystems, 12, 3, Q0AE02, doi:10.1029/2010GC003402.

Wernicke, B., Burchfiel, B.C., 1982, Modes of extensional tectonics: Journal Structural Geology 4: 105-115.

Wildman, R.A., Gazonas, G.A., 2009, Gravitational and magnetic anomaly inversion using a tree-based geometry representation: Geophysics, 74, I23–I35.

Wilsher, W.A., 1987, A structural interpretation of the Witwatersrand basin through the application of automated depth algorithms to both gravity and aeromagnetic data, M.Sc. thesis, University of Witwatersrand.

Won, I.J., Bevis, M., 1987, Computing the gravitational and magnetic anomalies due to a polygon: Algorithms and Fortran subroutines: Geophysics, 52, 232-238.

Yao, C., Hao, T., Guan, Z., Zhang, Y., 2003, High-speed computation and efficient storage in 3-D gravity and magnetic inversion based on genetic algorithms: Acta Geophysica Sinica, 46, 252-258.

Zervos, F., 1987, A compilation and regional interpretation of the northern North Sea gravity map: in Continental Extensional Tectonics, Eds. Coward, M.P., Dewey, J.F.., and Hancock, P.L., 28, 477–493.

Zhang, J., Zhong, B., Zhou, X., Dai, Y., 2001, Gravity anomalies of 2-D bodies with variable density contrast: Geophysics, 66, 809-813.

Zhang, C., Mushayandebvu, M.F., Reid, A.B., Fairhead, J.D., Odegard, M.E., 2000, Euler deconvolution of gravity tensor gradient data: Geophysics, 65 512–520.

Zhou, X., 2008, Two-dimensional (2D) vector gravity potential and general line integrals for gravity anomaly due to 2D masses of depth-dependent density contrast: Geophysics, 73, I43–I50.

Zhou, X., 2009, General line integrals for gravity anomalies of irregular two-dimensional (2D) masses with horizontally- and vertically-dependent density contrast: Geophysics, 74, I1–I7.

Zhou, X., 2013, Gravity inversion of 2D bedrock topography for heterogeneous sedimentary basins based on line integral and maximum difference reduction methods: Geophysical Prospecting, 61, 220–234.