"2.75D and 3D Automatic Modelling and Inversion Techniques of Gravity Anomalies using Variable Density-Depth Model: Space Domain Algorithms "

ORIGINALITY REPORT

21%	15%	20%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source Certified that it is student's publication, in which he is first author. (V. CHAKRAVARTHI)	10%
2	K. Mallesh, V. Chakravarthi, B. Ramamma. "3D Gravity Analysis in the Spatial Domain: Model Simulation by Multiple Polygonal Cross-Sections Coupled with Exponential Density Contrast", Pure and Applied Geophysics, 2019 Publication Certified that it is student's publication, in which he is first author. (V-CHAKRAVARTHI)	3%
3	academic.oup.com Internet Source	2%
4	Submitted to University of Hyderabad, Hyderabad Student Paper	1%
5	V. Chakravarthi, S. Rajeswara Sastry, B. Ramamma. "MODTOHAFSD — A GUI based JAVA code for gravity analysis of strike limited sedimentary basins by means of growing bodies with exponential density contrast–depth	1%

variation: A space domain approach", Computers & Geosciences, 2013
Publication

---

6 V. Chakravarthi, M. Pramod Kumar, B. Ramamma, S. Rajeswara Sastry. "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, 2016 — 1%
Publication

---

7 V Chakravarthi, K Mallesh, B Ramamma. "Basement depth estimation from gravity anomalies: two 2.5D approaches coupled with the exponential density contrast model", Journal of Geophysics and Engineering, 2017 — 1%
Publication

---

8 Vishnubhotla Chakravarthi, Batta Ramamma. "Determination of Sedimentary Basin Basement Depth: A Space Domain Based Gravity Inversion using Exponential Density Function", Acta Geophysica, 2016 — <1%
Publication

---

9 V. Chakravarthi, B. Ramamma, T. Venkat Reddy. "Gravity anomaly modeling of sedimentary basins by means of multiple structures and exponential density contrast- — <1%

depth variations: A space domain approach", Journal of the Geological Society of India, 2013
Publication

---

10 V. Chakravarthi, S. Rajeswara Sastry. "GUI based inversion code for automatic quantification of strike limited listric fault sources and regional gravity background from observed Bouguer gravity anomalies", Journal of the Geological Society of India, 2014 <1%
Publication

---

11 Chakravarthi, V.. "TODGINV-A code for optimization of gravity anomalies due to anticlinal and synclinal structures with parabolic density contrast", Computers and Geosciences, 200808 <1%
Publication

---

12 G. Parthasarathy. "Mathematical Analysis of Gravity Anomalies: Concepts, Algorithms and Computer Programs by V. Chakravarthi", Journal of the Geological Society of India, 2011 <1%
Publication

---

13 V. Chakravarthi, H.M. Raghuram, S.B. Singh. "3-D forward gravity modeling of basement interfaces above which the density contrast varies continuously with depth", Computers & Geosciences, 2002 <1%
Publication

14	V. Chakravarthi, S. Rajeswara Sastry, M. Pramod Kumar. "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, 2014 Publication	<1%
15	Chakravarthi, . "Gravity anomalies of pull-apart basins having finite strike length with depth dependent density: a ridge regression inversion", Near Surface Geophysics, 2009. Publication	<1%
16	Vishnubhotla Chakravarthi, Narasimman Sundararajan. "3D gravity inversion of basement relief — A depth-dependent density approach", GEOPHYSICS, 2007 Publication	<1%
17	V. Chakavarthi, N. Sundararajan. "Gravity Anomalies of 2.5-D Multiple Prismatic Structures with Variable Density: A Marquardt Inversion", Pure and Applied Geophysics, 2006 Publication	<1%
18	www.ria.ie Internet Source	<1%
19	V. Chakravarthi. "Automatic gravity optimization of 2.5D strike listric fault sources with analytically defined fault planes and depth-	<1%

dependent density", GEOPHYSICS, 2011
Publication

20	D. Bhaskara Rao, N. Ramesh Babu. "Three-dimensional analysis of gravity anomalies of sedimentary basins by polygonal prismatic model with a quadratic density function", Pure and Applied Geophysics PAGEOPH, 1993 Publication	<1%
21	David Wagg, Simon Neild. "Nonlinear Vibration with Control", Springer Science and Business Media LLC, 2015 Publication	<1%
22	moam.info Internet Source	<1%
23	Chakravarthi, V.. "INV2P5DSB-A code for gravity inversion of 2.5-D sedimentary basins using depth dependent density", Computers and Geosciences, 200705 Publication	<1%
24	rd.springer.com Internet Source	<1%
25	Giovanni Florio. "The Estimation of Depth to Basement Under Sedimentary Basins from Gravity Data: Review of Approaches and the ITRESC Method, with an Application to the Yucca Flat Basin (Nevada)", Surveys in	<1%

Geophysics, 2020
Publication

26 Chakravarthi, . "Gravity anomaly modelling of multiple geological sources having differing strike lengths and arbitrary density contrast variations", Near Surface Geophysics, 2013.
Publication

<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		

2.75D and 3D Automatic Modelling and Inversion Techniques of Gravity Anomalies using Variable Density-Depth Model: Space Domain Algorithms

A Thesis submitted during the year 2020 to the University of Hyderabad
in partial fulfilment of the award of a Ph.D. degree in
Centre of Earth, Ocean and Atmospheric Sciences, School of Physics

by
K. Mallesh

Centre for Earth, Ocean and Atmospheric Sciences
School of Physics

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad – 500 046
Telangana
India

CERTIFICATE

This is to certify that the thesis entitled '*2.75D and 3D Automatic Modelling and Inversion Techniques of Gravity Anomalies using Variable Density-Depth Model: Space Domain Algorithms*' submitted by *K.Mallesh,* bearing registration number *15ESPE03,* in partial fulfilment of the requirements for award of Doctor of Philosophy in *Geophysics* is a bonafide work carried out by him under my supervision and guidance.

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.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma.

Further, the student has the following publication(s) before submission of the thesis for adjudication and has produced evidence for the same in the form of reprints in the relevant area of his research: (Note: at least one publication in a refereed journal is required)

1. Chakravarthi, V., Mallesh, K., Ramamma, B., 2017, Basement depths estimation from gravity anomalies: Two 2.5D approaches coupled with exponential density contrast model, Journal of Geophysics and Engineering, 20, 303-315.
   Chapter of dissertation where this publication appears (Delete if not applicable) 2 & 3.
2. Mallesh, K., Chakravarthi, V., Ramamma, B., 2019, 3D Gravity Analysis in Spatial Domain: Model Simulation by Multiple Polygonal Cross-Sections coupled with Exponential Density Contrast: Pure and Applied Geophysics, 176(6), 2497-2511.
   Chapter of dissertation where this publication appears (Delete if not applicable) 4 & 5.

Further, the student has passed the following courses towards fulfilment of the coursework requirement for Ph.D.

Sl No.	Course Code	Course Title	Credits	Pass/Fail
1	ES 801	Earth System Sciences	4	Pass
2	ES 805	Research Methodology	3	Pass
3	ES 806	Mathematics for Earth Sciences	4	Pass
4	AP 811 – 830	Special Paper on Specified Research Topic	2	Pass
5	ES 807	Interdisciplinary course	3	Exempted

11 Dec 2020

Supervisor

Dr. V. Chakravarthi
Professor
Centre for Earth, Ocean & Atmospheric Sciences
University of Hyderabad
Hyderabad-500 046, Telangana, INDIA.

17 Dec, 2020

Head of the Centre

Head
Centre for Earth, Ocean & Atmospheric Sciences
University of Hyderabad
Hyderabad-500 046, INDIA.

Dean of School

DEAN
School of Physics
University of Hyderabad
Hyderabad-500 046, INDIA

Declaration

Centre for Earth, Ocean and Atmospheric Sciences
School of Physics
University of Hyderabad
Hyderabad – 500 046

I hereby declare that, this thesis entitled "*2.75D and 3D Automatic Modelling and Inversion Techniques of Gravity Anomalies using Variable Density-Depth Model: Space Domain Algorithms*" is the result of investigation carried out by me in the Centre for Earth, Ocean and Atmospheric Sciences (School of Physics), University of Hyderabad, India under direct guidance and supervision of Prof. V. Chakravarthi, is a bonafide research work and free from plagiarism. I also declare that it has not been submitted previously, in part or in full to this University or any other University or Institute, for the award of any degree or diploma.

A report on the plagiarism statistics from the University of Hyderabad Librarian is enclosed.

Name: K. Mallesh
Reg. No: 15ESPE03

Place: Hyderabad
Date: 11-12-2020

Acknowledgements

It is with a deep sense of indebtedness and respect that I thank my research supervisor Prof. V. Chakravarthi, Centre for Earth, Ocean and Atmospheric Sciences (CEOAS), University of Hyderabad for his expert guidance and constant encouragement throughout the progress of my research work.

I owe special thanks to Prof. M. Jayananda and Dr. Srilakshmi, members of my Research Advisory Committee (RAC), for their useful suggestions during the course of my research work.

Head, CEOAS is acknowledged for providing necessary facilities in the Centre to carry out my research work. I also thank my fellow research scholars, and particularly the supporting staff of CEOAS for their benevolent attitude.

I sincerely acknowledge University Grants Commission (UGC), Government of India for extending financial support in the form Rajiv Gandhi National Fellowship to pursue my research in the University.

Words fall short to express my heartfelt gratitude to my parents in particular, and other family members for their caring support and encouragement all through my career.

***

Preface

Geophysical methods play vital and significant roles not only in the search for concealed natural resources but also in addressing a wide spectrum of intrinsic problems associated with tectonics and geodynamics, engineering and environment, glaciology, volcanology, archaeology etc. Meaningful contrasts in physical properties such as density, magnetization, electrical resistivity/conductivity, elastic properties etc. between the source of interest and the host rock could generate corresponding anomalous field, which can be measured by properly deploying geophysical sensors on either the topography or in boreholes. These measured quantities are corrected to remove/suppress unwarranted signals before being subjected for analysis to quantify subsurface geologic structures.

The success of gravity method, which is the subject matter of this thesis, is largely dependent on considerable density contrast between the source of interest and the surrounding formations, which could generate a favorable gravity anomaly. Although the subsurface density distribution of sedimentary rocks is quite complex to explain; in most of the geological settings, the density of sedimentary rocks is proved to vary predominantly with depth. However, this advantage of this density function is offset by the fact that closed form analytical equations could not be derivable in the spatial domain to realize forward modeling. Although, closed form expressions could be derivable in the wavenumber domain for gravity anomalies, large truncation errors would pop-up in the anomalies when transformation of anomalies takes place from frequency to the spatial domain. Over and above, many sedimentary basins on the continental platform are strike limited. Hence, it imperative to develop new algorithms in the spatial domain to analyze the gravity anomalies of sedimentary basins by treating the anomalous source as strike limited and in which the density contrast obeys exponential variation with depth.

In this thesis, new forward modeling schemes are formulated in the spatial domain by combining both analytic and numeric approaches to derive anomaly expressions of strike limited sedimentary basins, among which the density contrast obeys exponential decrease with depth. The geometries considered in this thesis to describe sedimentary basins are, i) 2.75

dimensional (a special case of which is 2.5D) arbitrary shaped polygon, and ii) three dimensional (3D) polygon. Based on the principles of automatic modeling and inversion, new algorithms and related software, coded in JAVA, are developed in the spatial domain to analyze the gravity anomalies produced by sedimentary basins, treating the source independently as 2.75D and 3D. The applicability of each technique is demonstrated on both synthetic and real field gravity anomalies. In case of synthetic examples, anomalous field added with pseudorandom noise is considered as observed anomaly before being subjected to analysis. The inferred results are then examined against the assumed structures. In case of real field anomalies, the interpretations yielded by present techniques are compared with those obtained from drilling/reported information.

Accordingly, the thesis under study is organized into five chapters as detailed below.

In chapter-I, general introduction on the gravity method, gravity potential and gravity field, concept and role of exponential density model (EDM) to explain the density variation of sedimentary rocks, reasons to consider EDM over other density functions in vogue, and a brief account of work done so far on modeling and inversion are presented. Unlike the existing practice, the terms "automatic modeling" and "inversion" are distinguished from each other by formulating a criteria. The same criteria is followed throughout the thesis to design automatic algorithms to analyze the gravity anomalies of sedimentary basins.

In Chapter II, a new forward modeling scheme is developed in the spatial domain using EDM to compute the gravity anomalies of 2.75D sedimentary basins. The effect of offset of the profile (i.e., when the profile fails to bisects the strike length of the anomalous source) on the magnitude of the anomalous field is explained in length. The principle of automatic modeling is used to develop an interpretation technique, again in the spatial domain, to analyze the gravity anomalies to recover basement depths. A GUI based software POLYMODEXP, coded in JAVA, is developed based on the interpretation methodology to analyze the anomalies. This code, operates on Model-View-Controller (MVC) framework. The advantage of this software is that it has an inbuilt graphical user interface, which enables the interpreter to visualize animated versions of i) the model growth at the end of each iteration, ii) corresponding improvement in the model response, ii) changes in the misfit, and iii) variation of density contrast with depth. The noteworthy advantage of the proposed method is that it can be

implemented to analyze the anomalies even when the profile fails to bisects the strike length of the basin. Efficiencies of the modeling scheme and software are illustrated with the gravity anomalies of a theoretical model in the presence of pseudorandom noise. The real field gravity anomalies across the Gediz and Büyük Menderes grabens in western Turkey using the derived EDMs have produced geologically reasonable results which are in close agreement with those reported previously.

A gravity optimization strategy (built on the principles of inversion) and related software, POLYINVEXP, coded in JAVA to analyze the gravity anomalies produced by 2.75D sedimentary basins using EDM are presented in Chapter- III. This inversion strategy, unlike the case of modeling, analyzes the observed anomalies for estimating both basement depths and regional gravity background simultaneously. Forward gravity modeling is realized in the spatial domain with appropriate analytic and numeric approaches. The MVC based software, POLYINVEXP, reads the Bouguer gravity anomalies, constructs and modifies regional gravity background in an interactive approach, and performs business logic to recover basement topography. In addition to generating the output in ASCII and graphical forms, the software also displays the animated versions of model space improvement and corresponding changes in model gravity response with the iteration number. The efficiency of inversion is demonstrated with the same noise gravity anomalies used in case of automatic modelling in the presence of regional gravity background. It was found from the analysis that the estimated structure from inversion closely mimics the structure that was inferred with automatic modelling. Inversions performed over the gravity anomalies across the Gediz and Büyük Menderes grabens in western Turkey using the derived EDMs have yielded models that are more or less similar with the estimated structures from automatic modeling. Over and above, the estimated regional in case of synthetic example closely mimics the assumed regional.

Based on the principles of modeling, an automatic 3D technique coupled with GUI based software, POLYMOD3D (coded in JAVA), is developed using EDM to model the gravity anomalies arise from 3D sedimentary basins and presented in chapter-IV. The sedimentary column above the interface is described with a stack of multiple vertical polygonal sections of unit thickness each. For such a case, the depth ordinates of the vertices of the cross-sections become the unknown parameters to be estimated from gravity data. Forward solution of the

model space is realized in the spatial domain by a technique that combines both analytic and numeric approaches. Initial depths to the interface are calculated based on the Bouguer slab approximation and subsequently improved, iteratively, based on the ratio of the product of the observed gravity anomaly and existing depth parameter to the corresponding model gravity response. The iterative process continues till one of the predefined termination criteria is accomplished. The code, POLYMOD3D, works on MVC pattern, reads the residual gravity anomalies of a sedimentary basin and estimates basement depths at plurality of locations. 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 applicability of the proposed modeling is exemplified with a set of noisy gravity anomalies attributable to a synthetic structure before being applied to a real world gravity data. In the case of the synthetic example, the method has yielded a structure that was compatible with the assumed structure even in the presence of random noise. Application of the proposed method to the gravity data set from the Los Angeles basin, California using a prescribed exponential density model has yielded a model that concurs well with the published models.

In Chapter-V, A3D inversion technique and related software, POLYINV3D (coded in JAVA) using the principles of Ridge Regression algorithm is developed in the spatial domain to estimate basement depths of sedimentary basins from a set of observed gravity anomalies using a prescribed exponential mass density contrast. Initial depths to the basement at plurality of observations are calculated presuming that the density contrast within the slab also varying exponentially with depth. The approximate depths are updated, iteratively, by adding/subtracting the increments/decrements in model depths that are obtained from the solution of normal equations within specified convergence criteria. The software, POLYINV3D, 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. Recovery of basement depths with modest errors from a set of noisy gravity anomalies attributable to a synthetic model and also the fact that the estimated depth structure of the Almaz*á*n Basin in northeast Spain from the observed gravity anomalies correlates well with the

information derived from seismic studies demonstrates the applicability of the present inversion method. The inversions when performed on the gravity anomalies considered in the previous Chapter-IV have 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 softwares' presented in this thesis are platform independent.

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

00--000--00

List of figures

List of Tables

Contents

CHAPTER
ONE

Introduction

1.1 General

Though applied Geophysics is relatively a new branch of science, much has happened last over a few decades particularly in areas of design and development of sophisticated geophysical instruments, planning and execution of field operations, processing and interpretation of geophysical data, development of sophisticated mathematical tools and related software etc. The main task of geophysics is to determine, as much as possible, about the internal structure and dynamics of the earth, concealed objects, utilities and geologic structures that hold mineralization etc. from a set of geophysical measurements collected over the topography or in boreholes or at times a combination of both.

A wide spectrum of geophysical techniques are in vogue, which rely on specific physical property contrasts of the subsurface objects to yield corresponding anomalous fields. The inferences drawn from surface geophysical measurements such as gravity, magnetic, electrical, electromagnetic, seismic, radiometric etc. would unambiguously provide crucial information to comprehend the complexities of the subsurface geology. Through the systematic application of the laws of physics, powerful mathematical and statistical tools and computational infrastructure geophysical interpretations definitely would cater sensible solutions to a range of geologic problems at different operational scales. In principle, variations in physical fields are being measured in an area of investigation, and the observed signals are corrected for unwarranted signals/factors so as to obtain the information attributable to target source(s). These corrected geophysical signals/anomalies are then interpreted in terms of either specific geometry(ies) or physical property contrast(s) and in some special cases a combination of both. However, interpretation of geophysical signals measured on the surface topography is generally non-unique owing to the simple reason that the surface measured signals are the aggregate

effects of different signals originating from different depths and also noises of different origin. Such abstruseness and indecision in geophysical analysis can be effectively abridged to a large degree by properly handling the data sets, besides using additional information obtained from surface outcrops, boreholes, other geophysical methods etc. (Chakravarthi et al., 2007).

In the method of gravity exploration the innately arising changes in earth's gravity field are studied to explore the subsurface for a variety of objectives. Newton's law which is the basis for the gravity work can be stated that the gravitational force, F, between two linearly separated point masses m_1 and m_2 is directly proportional to the mass product and inversely to the distance square, r^2, between them i.e.,

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

Here G is universal gravitational constant. The negative sign preceding the right hand term of equation (1.1) suggests that the gravitational force all the time is attractive. If m_1 is the mass of the earth, M_e the acceleration, g, of m_2 at the surface of the earth can be expressed as

$$g = \frac{F}{m_2} = \frac{-GM_e}{R_e^2}, \tag{1.2}$$

where, R_e is the radius of the earth. The force experienced by unit mass on the earth's surface is referred to as the gravity acceleration. This is also referred to as the gravity field of the earth, which varies from 978.0327 Gals from equator to 983.2186 Gals at the poles (Wilcox, 1989; Chakravarthi, 2011a; Chakravarthi, 2020).

The gravitational force is a vector; giving rise to a conservative field that can be derived from scalar potential as

$$\Delta U(r) = \frac{F(r)}{m_2} = g(r) \tag{1.3}$$

Alternatively, this equation can be solved for the gravity potential as

$$U(r) = \int_{\infty}^{R} g\, dr = -GM_e \int_{\infty}^{R} \frac{dr}{r^2} = \frac{GM_e}{R}. \tag{1.4}$$

Gravity problems can easily be solved by calculating the scalar potential, U, rather than vector, g. The potential and the acceleration of gravity at a point, P, away from a distance, r, can be calculated by dividing the mass into number of elements and then integrating the respective contributions as

$$dU = G\frac{dm}{r} = \frac{G\sigma(dxdydz)}{r}, \tag{1.5}$$

where, σ is the density, and $(dxdydz)$ represents the volume of a small element within the source. The coordinates of the element are given by (x, y, z), and r is the distance vector connecting the origin and the element, whose magnitude is given by $r^2 = x^2 + y^2 + z^2$.

The total potential of the mass, m, can be worked out as

$$U = G\sigma \iiint_{v} \frac{dxdydz}{r}. \tag{1.6}$$

The gravity acceleration along the z-axis (considered positive vertically downwards) can be obtained as

$$g_z = -\frac{\partial U}{\partial z} = G\sigma \iiint_{v} \frac{dxdydz}{r^3}. \tag{1.7}$$

If a source is considered of having theoretically infinite strike along the y-axis with uniform cross-section throughout, then the gravity potential at the origin can be calculated as

$$U = 2G\sigma \iint_{s} \log\left(\frac{1}{r}\right) dxdz\,. \tag{1.8}$$

Here, $dxdz$ is the area of cross-section of a solenoid. The gravity contribution of the said 2D source can be obtained as

$$g_z = -\frac{\partial U}{\partial z} = 2G\sigma \iint_S \frac{z dx dz}{r^2}, \tag{1.9}$$

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

Equation (1.7) can used to calculate the gravity effects of 2.5D, 2.75D, and 3D anomalous sources, and equation (1.9) to calculate anomalous fields due to 2D sources. The lateral contrast in density between the target source and the surroundings is exploited to measure the corresponding anomalous gravity field either or both on the surface of the earth as well as in boreholes. The measured gravity field is subsequently corrected for the unwarranted effects that arise due to the earth's topography, the nature of the subsurface etc. before being attempted for quantitative interpretation followed by geological translation. Unequivocally, a thorough knowledge on densities of different rock formations is always a prerequisite for efficient modeling of gravity anomalies to untangle subsurface geologic complexities.

1.2 Exponential Density Model (EDM)

In case of sedimentary rocks, it is confirmed beyond doubt that the density varies non-uniformly with increasing depth because of several factors such as degree of tectonic disturbance, diagenesis, facies change, stratigraphic layering, compaction due to geostatic pressure, cementation etc.

Based on the study of more than 2200 rock samples collected from deep boreholes located in north eastern Oklahoma and Texas, Athy (1930) had demonstrated that the variation in density of different types of sedimentary rocks against the depth of burial could be affectively modeled by an exponential equation. With a collection of more than 900 measured samples on porosities and densities of different sedimentary rocks across the globe, Manger (1963) had shown that the density of sandstone and pure shale formations increases with depth, whereas their porosity decreases. From the measurements of bulk density, grain density and porosity of the rock samples recovered from a deep well of about 1300m depth in the Indus Cone of the Arabian Sea, Bachman and Hamilton (1976) reported that the density of marine sediments increases sharply from ~1.58 g/cm^3 near the ocean bottom to ~2.29 g/cm^3 at 1200m depth. From an exhaustive analysis of 435 measured density logs from deep-drilled boreholes in Western

Canada, Maxant (1980) had demonstrated that many sedimentary rocks did not show any linear relationship with increasing depth. Dimitropoulos and Donato (1981) had reported based on the well data that the average density of younger Cretaceous sediments in the Inner Moray Firth sedimentary basin, Scotland was 2.32 g/cm^3, whereas the density of underlying older Jurassic and Permo-Triassic sediments were 2.44 g/cm^3 and 2.52 g/cm^3, respectively. Nelson and Fairchild (1989) had shown from the Gulf of Maxico that the sedimentary rocks in this region portray rapid increase in density within the top 1.5 km. The collected density data from a deep borehole in a 9 km thick-sectioned sediment from the Viking graben (Donato and Tully, 1981; Zervos, 1987), Cowie and Karner (1990) showed that the average density of sedimentary rocks explicitly portray an increasing trend with the depth though the sedimentary rocks densities vary over a wide range. Such typical behavior of sediment density with increasing depth is also reported by Chakravarthi and Pramod Kumar (2015a) from the Chintalpudi sub-basin, India. On the other hand, Dickinson (1953), Dallmus (1958), Storer (1959), McCulloh (1967), Eaton (1969), Rieke et al. (1974), Castagna et al. (1993) have elucidated that the assessed density-depth profiles of shale formations from different sedimentary basins across the globe in different geologic settings and depositional environments showed similar behavior.

Using the drill hole density samples from the North German-Polish basin, Schön (1996) had shown that the younger sedimentary rocks portray rapid increase in density at shallower depths when compared to older sedimentary rocks. Mooney and Kaban (2010), based on the drill-hole log data, from the Michigan and Illinois basins, showed that sedimentary rocks density varies less significantly on the continental regions when compared to offshore basins. Based on the study of the measured density samples collected from 716 drill sites, Tenzer and Gladkikh (2014) explained that a non-linear relationship exists between the densities of marine sediments with depth, while Gu et al. (2014) opined that significant density contrast exists between the overlying sediments and the underlying basement in case the thickness of sedimentary pile is minimum.

Li (2001) opined that the subsurface density distribution of sedimentary rocks is often quite complex to explain. Cai and Zhdanov (2015) opine that, in general, the density-depth data of sedimentary rocks does not follow any specific mathematical expression owing to the fact that several geologic factors and diversified tectonic settings influence the density. Nevertheless,

Cordell (1973) had illuminated with field examples that the sediment density primarily increases with increasing depth, more sharply at shallower depths than at deeper depths and such variation could be parroted many times by simple mathematical functions. In this context, several functions have been coined to describe the density/density contrast variation of sedimentary rocks with depth, viz., linear (Pedersen 1985; Pohanka, 1988; Reamer and Ferguson 1989; Hansen, 1999; Holstein, 2003; Hamayun et al., 2009; Prutkin and Tenzer, 2009; D'Urso, 2014a; D'Urso, 2015a; Lin and Danker, 2019; Chen et al., 2019a), quadratic (Rao, 1985; Rao, 1986; Murthy et al., 1989; Rao and Prakash, 1990a; Rao et al., 1990b; Rao, 1990c; Martín- Atienza and García-Abdeslem, 1999; Kadirov, 2000; Zhang et al., 2001; Gallardo-Delgado et al., 2003; Gallardo et al., 2005; Nasuti and Ardestani, 2007; Zhou, 2010; D'Urso, 2015b; Feng et al., 2015), cubic (García-Abdeslem, 2003; García-Abdeslem, 2005, Ren et al., 2017), exponential (Cordell, 1973; Granser, 1987; Chai and Hinze, 1988, Xia and Sprowl, 1992; Rao et al., 1993; Rao and Rao, 1999; Szabo and Pancsics, 1999; Engen et al., 2006; Zhou, 2008; Chappel and Kusznir, 2008; Wang et al., 2011; Chakravarthi et al., 2013a; Chakravarthi et al., 2013b; Chakravarthi and Ramamma, 2015b; Feng et al., 2015; Chakravarthi et al., 2016; Chakravarthi et al., 2017a; Chakravarthi et al., 2017b; Pham et al., 2018, Mallesh et al., 2019; Wu, 2019; Chen et al., 2019b; Ramamma et al., 2020).

The efficacy of the above-listed density functions namely, linear, quadratic, cubic and exponential to explain the density contrast variation of sedimentary rocks is demonstrated with a synthetic density log shown in Fig. 1.1 (Ramamma et al., 2020). In this case, it is presumed that density contrast-data of sedimentary rocks is available up to a maximum logging depth of 2.0 km (step line in Fig. 1.1a) against the total depth of 4.0 km. The utility of the above density functions to describe the density contrast variation is examined by fitting the density functions to the available density contrast-depth data. For having clarity, the simulated density models are presented in two depth windows i.e., from 0 km to 2.0 km (Fig. 1.1a), and 2.0 km to 4.0 km (Fig. 1.1b). The linear, quadratic and cubic density models are defined as

$$\Delta\rho(z) = \sum_{\zeta=0}^{n} b_\zeta z^\zeta, \begin{cases} n = 1 \; for \; linear \\ n = 2 \; for \; quadratic \\ n = 3 \; for \; cubic, \end{cases} \tag{1.10}$$

where, $\Delta\rho(z)$, is the density contrast at any specific depth, z. Here, b_ζ, represents a set of coefficients of respective density models. On the other hand, the exponential density model, $\Delta\rho(z)$, is defined as (Cordell, 1973; Chakravarthi et al., 2013a; Mallesh et al., 2019; Ramamma et al., 2020)

$$\Delta\rho(z) = \Delta\rho_0 e^{-\lambda z}, \tag{1.12}$$

where, $\Delta\rho_0$ is the density contrast at the surface, and λ is a decay constant expressed in inverse length units.

The estimated coefficients of the fitted density models, under consideration, are given in Table 1.1 and shown graphically in Fig. 1.1.

Table 1.1
Derived coefficients of density models (after Ramamma et al., 2020)

Density model	**Coefficients**			
	b_0	b_1	b_2	b_3
LDM	-0.6611735	0.3698231		
QDM	-0.8084931	0.8625286	-0.2568058	
CDM	-0.8476584	1.1180584	-0.5854998	0.10962532
	$\Delta\rho_0$	λ		
EDM	-0.916764	1.5262346		

LDM: Linear Density Model
QDM: Quadratic Density Model
CDM: Cubic Density Model
EDM: Exponential Density Model

It can be noticed from Fig. 1.1a that the exponential (EDM), quadratic (QDM) and cubic (CDM) density models are more effective than the linear density model (LDM) to describe the density contrast variation up to a logging depth of 2 km. However, beyond 2 km depth all the density functions except the EDM portray undue deviations with increasing depth (Fig. 1.1b). For e.g., LDM and CDM show unusual density reversals (positive density contrasts) with increasing depth, whereas QDM results in large deviations from the expected trend both in magnitude and direction (Fig. 1.1b).

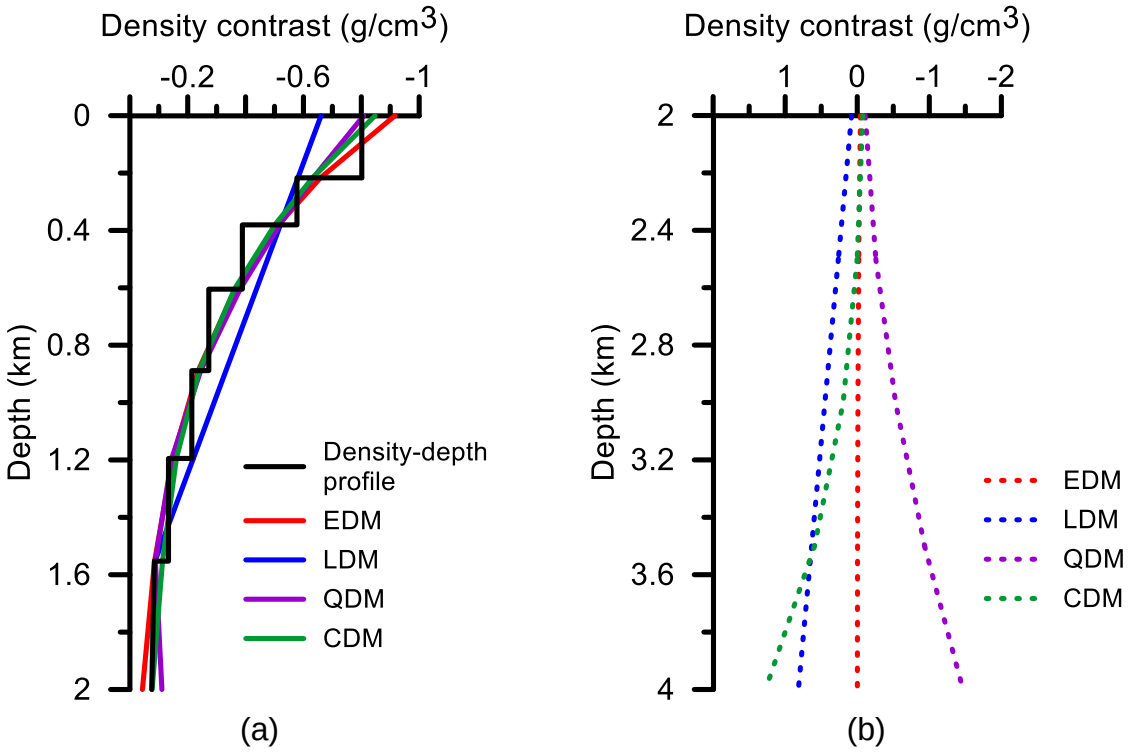


Fig. 1.1 (a) Assumed (step line) versus simulated density models within the logging depth of 2 km (solid lines in color), and (b) extrapolated density models (dashed lines in corresponding color) beyond the logging depth of 2 km (after Ramamma et al., 2020).

The above study unambiguously demonstrates that LDM, QDM and CDM do not succinctly explain the true density variation at deeper depths. It is to be realized that in exploration programs for natural resources geophysical well-logs are generally limited to modest depths from the probing boreholes (Chakravarthi and Sundararajan, 2006a), and in such cases, LDM, QDM and CDM find rather limited practical use to accurately model the complete density-depth profile. It is obvious to note that even higher order polynomials (Wu and Chen, 2016; Jiang et al., 2017; Zhang and Jiang, 2017; D'Urso and Trotta, 2017; Ren et al., 2017; Chen et al., 2018; Chen et al., 2019; Wu, 2018; Fukushima, 2018; Liu et al., 2019) would become ineffective in such scenarios. On the other hand, the characteristic increase in sedimentary rock density - more rapidly at shallower depths and less progressively at deeper depths (Cordell, 1973) could be affectively explained by an exponential law given by equation (1.12). The characteristic features of the anomalous gravity field produced by a strike-limited basin with the enlisted density models is discussed in Chapter-2.

Though the exponential density model is far superior when compared to other density functions to describe the density variation with depth, this advantage is slightly offset by the fact that it does not allow one to derive closed-form analytical gravity expressions to realize forward modeling of geologic structures in the spatial domain (Murthy and Rao, 1979; Rao and Murthy, 1989; Chakravarthi and Sundararajan, 2004b; Chakravarthi, 2011b; Chakravarthi et al., 2013b; Chakravarthi et al., 2017b; Mallesh et al., 2019). In this thesis, new forward modeling schemes are formulated in the spatial domain to calculate the gravity anomalies of 2.75D/2.5D, and 3D sedimentary basins in which the density contrast obeys exponential variation with depth. In all cases, solutions for anomaly equations are obtained in the spatial domain by judiciously combining analytic and numeric methods. The validity of each such expression to calculate the gravity anomalies is confirmed by judging the correlation between the anomalies obtained from the present technique with those of the analytic solution of respective homogenous models.

1.3 Quantitative interpretation

Quantification of gravity anomalies amounts to estimating appropriate model parameters of geophysical geometries using observed data-sets such that the derived model can explain the geology. However, as pointed out by Roy (1962), Backus and Gilbert (1967, 1968), Rao and Murthy (1978), and Blakely (1995) interpretation of gravity anomalies suffers from non-uniqueness and often ill-posed because of the fact that several density distributions can equally explain the observed anomalous field on the topography. Such ambiguity can be handled by making use of independent information (derived from known geology, boreholes and other geophysical methods) in interpretation while estimating unknown parameters of model space.

Quantitative methods of interpretation can be broadly categorized into i) direct methods, ii) indirect methods, and iii) data enhancement methods. The first category comes under the classical trial and error methods, wherein a simple model is constructed based on the knowledge of geology and its gravity effect is calculated and judged with measured anomaly. The mismatch between the observed and predicted anomalies is then reduced in an interactive mode by changing either parameters of the model or its density contrast or a combination of both. However, the utility of these methods is rather limited in reality because achieving a satisfactory fit between the observed and computed anomalies almost become a herculean task when the parameters to be estimated are too large in number. In the case of indirect methods, a model is

initiated by the interpreter or in some specific cases by the computer itself based on a few characteristic anomalies and the model is subsequently updated in an iterative mode following some predefined fitting criteria. In doing so, bounds are generally imposed on the unknown parameters in order to restrict the range of possible admissible solutions. On the other hand, data enhancement methods do not estimate the model parameters as such from the gravity anomalies but could enhance some features of the anomalous source (Blakely, 1995).

In indirect methods, interpretation of gravity anomalies is performed in either physical property mode or geometric mode. In physical property mode, the model space is discretized into a large number of cells, generally, of equal size and interpretation is performed for estimating the densities of cells (Medeiros and Silva, 1996; Li and Oldenburg, 1998; Ekinci, 2008; Liu et al., 2015; Rezaie et al., 2017); whereas in the geometric mode, optimum parameters of prescribed geometries are estimated by specifying the density contrast/s as a known parameter (Murthy, 1998; Ramamma et al., 2020 and references there on). Among these two approaches, the geometric mode of interpretation is more popular because the number of parameters to be estimated as well as the model uncertainty would be reduced to a large extent (Cai et al., 2018; Mallesh et al., 2019; Ramamma et al., 2020). The interpretation algorithms presented in this thesis come under the geometric mode of interpretation.

Not much attention has been focused in the literature to distinguish automatic modeling schemes from that inversion. In this thesis, criteria advocated by Murthy (1998), Chakravarthi et al. (2013a), Chakravarthi et al. (2017a) has been followed to develop independent algorithms for automatic modeling and inversion strategies for analyzing sedimentary basins' gravity anomalies using predefined EDM. Though the aim of both strategies remains the same to find out the basement depths of hidden basement interfaces from measured gravity anomalies, but they differ from each other in their approach and application. For e.g., automatic modeling schemes make use forward difference approximation for model updating, whereas inversion schemes sought solution of normal equations. In automatic modeling schemes the mismatch between the measured and theoretical anomalies at any observation is used to improvise the depth parameter of the basin at that observation only, whereas in inversion the errors at all observations are used to simultaneously to estimate the improvements in unknown quantities. Besides, modeling schemes yield reliable interpretations only on residual anomalies alone,

whereas inversion strategies yield acceptable solutions even in the presence of specific regional component (Chakravarthi, 2009).

1.4 Review of existing methods

Interpretation of gravity anomalies by thumb rules (Smith, 1959; Smith, 1960; Skeels, 1963), characteristic curves (Grant and West, 1965; Rao et al., 1974; Rao et al., 1979; Rao and Murthy, 1981), logarithmic curves (Hutchison, 1958; Murthy, 1969), nomograms (Pal, 1983) etc. were well-known ever since the gravity method was adopted in exploration programs. However, with the advent of digital computing machines and powerful mathematical tools for processing and interpretation these methods are now almost become obsolete.

Many methods in space and frequency domains are developed to quantify subsurface anomalous structures from the anomalies observed over them. Techniques in the frequency domain based on the application of Fourier transform were developed by Odegard and Berg (1965), Sharma and Geldart (1968), Meinardus (1970); Collins et al. (1974), Bhattacharyya and Leu (1975), Sengupta, (1977); Sharma and Bose (1977), Regan and Hinze (1977, 1978); Mohan (1978), Bhimasankaram et al. (1977), Murthy and Rao (1980), Chacko and Battacharya (1980), Pal (1981a, 1981b), Ruotoistenmäki (1983), Mareschal (1985), Thorarinsson et al. (1988), Rao et al. (1993), Annecchione et al. (2001), Rao and Avasthi (2006), Bhimasankaram et al. (2006), Phillips (2014), Dai et al (2019). Methods based on the Hilbert transform (Sundararajan et al., 1983; Ashena and Ardestani, 2012), Mellin transform (Mohan et al., 1986), Mellin-Fourier transform pair (Mohan et al., 1989), Walsh transform (Shaw and Agarwal, 1990, Keating, 1992; Al-Garni, 2008), Hartley transform (Sundararajan and Brahmam, 1998, Li et al., 2011), Sundararajan transform (Sundararajan et al., 2000; Al-Garni et al., 2010); hyperbolic S-transform (Mousavi and Ardestani, 2016), wavelet transforms (Bovanloo et al., 2012; Fedi and Mastro, 2018) are also available to analyze the anomalies of some simple geophysical geometries. The real difficulty associated with many transform methods is that a thorough knowledge on source depth is required to define optimum window size to analyze the anomalies (Chávez et al., 1999). Besides, significant errors would invariably crop up in depth estimates if the calculated spectrum becomes too complicated in the presence of multiple sources (Odegard, 2011; Fedi and Mastro, 2018).

The Euler deconvolution method is yet another method that has gained wide popularity in the interpretation of gravity anomalies for source depth estimations (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; Keating, 1998; Zhang, 2000; Roy et al., 2001; Hansen and Suciu, 2002; Gimenez et al., 2009; Thurston, 2010; Stavrev and Reid, 2010; Hu et al., 2011). However, the efficacy of this method is largely dependent on the selection of Source Structure Index (SSI), and that any deviation of specific SSI from its optimum value would lead to enormous errors in interpretations (Reid and Thurston, 2014; Reid et al., 2014). Furthermore, this method does not yield acceptable solutions if multiple bodies are present vertically one below the other (Barbosa and Silva, 2011). Lafehr and Nabighian (2012) opine that SSI that corresponds to a simple geometry (like sphere, sheet, cylinder, etc.) may not be adequate to deal with the anomalies of complex structures. Above all, the correctness of solutions derived from frequency domain methods as well as Euler deconvolution is always difficult to judge because they do not offer any direct means to compare theoretical anomalies with the observed ones.

In recent past, interpretation tools based on the application of neural networks are proposed to analyze the anomalies. These methods are very effective in providing viable solutions even when the misfit norm has several local minima. Osman et al. (2006), and Osman et al. (2007) have presented schemes using Forced Neural Network (FNN) to analyze anomalies assuming horizontal cylinders and prisms as sources. A 2D non-linear inversion method using the RBF (Radial Basis Function) neural network was proposed by Geng and Yang (2013) to decipher subsurface density distribution from gravity anomalies. Eshaghzadeh and Hajian (2018) have developed a Modular Feed-forward Neural Network (MFNN) method to find the parameters of simple shaped bodies from the measured anomalies. Eshaghzadeh et al. (2020) also have applied FNN to model the anomalies resulted from a vertical cylinder. Another group of global optimization methods, namely the genetic algorithms, employ the ideologies of biological evolution to search for the ideal solution in the entire model space pertaining to an inverse problem (Goldberg, 1989; Shi, 1992; Boschetti et al., 1997; Roy et al., 2002; Yao et al., 2003; Zhang et al., 2004; Bezada and Zelt, 2011). In Monte Carlo search methods the information derived from geophysical data is studied in combination with the available subsurface parameters using a posteriori probabilistic distribution to infer the optimum solution

(Mosegaard and Tarantola, 1991; Mosegaard and Tarantola, 1995; Tarantola, 2005; Ayala et al., 2019). The disadvantage of these methods is that they consume enormous amount time to populate model space as dense as possible to find the optimum solution for a given problem. Besides, these algorithms are effective with limited number of unknown parameters to be estimated.

Application of wavelet transforms for imaging structural boundaries from gravity data has gained accelerated attention among the geophysical community since late 1990s. The application of wavelet transforms in potential fields was documented by Moreau et al. (1997), Moreau et al. (1999). Using synthetic and real field gravity anomalies, Chapin (1997) demonstrated how the wavelet transforms could help in resolving interfering anomalies produced by closely spaced objects. Marlet et al. (2001) have applied 1-D wavelet transform on a gravity profile to delineate configurations of the major thrusts of Himalayas falling in Nepal. Using 2D continuous wavelet transform (CWT), Ouadfeul and Aliouane (2013a), and Ouadfeul and Aliouane (2013b) have successfully brought out the structural characteristics of the Hoggar region, and the Basin and Range Province in the United States from the gravity anomalies. Dogru and Pamukcu (2016) also have investigated the boundaries of discontinuities in the eastern part of Turkey from the gravity data using wavelet transform. A detailed review on the theory of continuous wavelet transform for analyzing potential field anomalies was documented by Sailhac et al. (2009).

Beiki (2010) had utilized the concept of analytic signal to gravity gradient tensor data by introducing analytic signal functions in three mutually perpendicular directions of the coordinate system. He had shown that the edges of subsurface anomalous targets can be better detected by computing the first vertical derivatives of the analytic signal in x and y directions. This method was proved to be insensitive to the interference effects of neighboring sources but significantly sensitive to the noise. Salem et al. (2013) have proposed an adaptive tilt angle method, which uses three vertical tensor components of full tensor gravity gradiometry data to locate objects that have simple geometric shapes. Again, this method is suitable to deal only with isolated sources besides the data should be completely noise free.

On the other hand, when interpretation is intended in the space domain (in geometric mode), the number of data generally far exceeds the number of unknown parameters. Such an over-determined problem can be effectively solved by least squares method (Lafehr and Nabighian, 2012). Because regularization is generally not required in geometric mode, the norm minimization contains only data misfit without model roughness. However, when interpretation is aimed in physical property mode the number of unknowns surpass the number data points, therefore, the misfit norm should invariably involve both data misfit and model roughness. By and large, the Gauss- Newton's method will be able to cater viable solutions for geometric mode inverse problems provided i) the initial model parameters are in close proximity to true parameters, ii) forward modeling operators are differentiable, and iii) the misfit norm is characterized by one minimum (Tarantola, 2005).

Mohan et al. (1986) had used the tunneling algorithm (Levy and Montalvo, 1985) in an inversion technique to analyze the gravity anomalies. Sastry and Moharir (1990) had in combination the tunneling algorithm and Tikhonov regularization to find the optimum solutions for gravity inverse problems. The tunneling algorithm operates in a recursive mode by traversing through a series of local minima of decreasing magnitudes and finally reaches to a desired minimum and the corresponding model space becomes the true solution. A detailed account on the application of Tunneling algorithm was given by Moharir (1990). Interpretation techniques using non-guided global optimization methods viz., simulated annealing (Mundim et al., 1998; Nagihara and Hall, 2001; Roy et al., 2002; Yu et al., 2007a, Yu et al., 2007b, Biswas, 2015; Biswas, 2016; Biswas et al., 2017), ant colony optimization (Gupta et al., 2011; Srivastava et al., 2013), particle swarm optimization (Toushmalani, 2013; Pallero et al., 2015; Singh and Biswas, 2015; Essa and Munschy, 2019) are also being used extensively in recent times to study the gravity signatures of idealized simple shapes. These techniques are, however, exorbitant in terms of memory requirements and computational time.

The assumption that anomalous sources possesses constant density in the enlisted methods is hardly ever applicable to analyze the gravity anomalies of the structures associated with sedimentary rocks for the reasons cited in section 1.2. Therefore, it is imperative to design new interpretation strategies and related software to analyze sedimentary basin gravity anomalies considering appropriate non-uniform density models.

1.5 Aim and scope of the thesis

The aims and objectives of this thesis are to develop interpretation algorithms based on the principles of automatic modeling and inversion to analyze the gravity anomalies produced by 2.75D/2.5D, and 3D sedimentary basins among which EDM models the density contrast variation with depth. This thesis also presents relevant software of the algorithms coded in core JAVA. The striking features of these codes are that they are platform independent and have GUI features inbuilt. Besides, these codes facilitate the user to customize different modules, if necessary. Accordingly, the thesis under study is structured into six chapters as follows.

In Chapter-1, general principles of the gravity method, fundamental relations connecting gravity potential and gravity field, use and abuse of existing density functions to simulate density-depth profiles of sedimentary rocks, general principles of quantitative interpretation of gravity anomalies, and a brief review on the existing interpretation methods are presented.

Chapter-2 deals with the development of an automatic modeling technique and associated software to interpret the anomalies of 2.75D/2.5D sedimentary basins with prescribed EDM. The interface between sediment and underlying basement is approximated by a finite-strike polygon with many vertices, and the ordinates of which become the unknown entities to be determined. Before that, the nature of gravity anomalies that are deemed to be produced by a typical finite-strike sedimentary basin in which the density contrast differs according to the formulations narrated in section 1.2 is discussed in length. It is also shown that the magnitude of anomalous field over a structure is strictly offset dependent, and use of said parameter in the analysis is necessary for reliable interpretations. Reliability and applicability of the methodology as well as the code are exemplified with a set of noisy gravity anomalies attributed to a synthetic model of sedimentary basin and also validated with the real field data pertaining to the Gediz and Büyük Menderes grabens of western Turkey. In case of field examples, EDM is derived from known density-depth information of respective grabens.

In Chapter-3, an interpretation technique using the principles of optimization is presented along with relevant software to interpret sedimentary basin gravity anomalies using EDM. This technique treats both basement depths and coefficients of left out regional gravity component (described by a linear equation) as unknown parameters and estimates the same. The geometry

of model space in this case remains the same as the one considered in automatic modeling. The algorithm uses the errors between measured and theoretical gravity anomalies and anomaly gradients with respect to the unknown parameters so as to construct as many normal equations as the number of unknown parameters and then solves the system for improvements in unknown parameters. The applicability of the proposed inversion is examined on synthetic as well as on real world gravity anomalies. The data sets used in Chapter-II under automatic modeling are analyzed in this chapter by inversion. The inversion results are compared with those of automatic modeling. In the case of synthetic example, the effectiveness of inversion is established by interpreting the anomalies of the structure considering both random noise and regional component.

In Chapter-4, a 3D technique coupled with relevant software to interpret sedimentary basin gravity anomalies with EDM is dealt with. Again this technique operates on the automatic modeling principles to recover basement configurations of basins from spatially distributed gravity data sets in automatic sense. A collage of multiple vertical laminas in xz plane along the strike of a sedimentary basin describes the model space. Each vertical lamina is treated as a polygonal cross-section with unit thickness and a prescribed EDM describes the density contrast variation within it. For such problem statement, the vertices of several representative polygons become the parameters to be solved from a given gravity data set. This technique initiates the model space and then updates it in iterative mode till one of the predefined convergence criteria satisfies. The technique was applied to recover the basement structure of a synthetic model from the gravity data set followed by analyzing the gravity anomalies collected over the Los Angeles basin, California. The interpreted results are compared with the assumed structure in case of the theoretical example and with the reported surface and sub-surface information in case of the field example.

Chapter-5 deals with the development of an automatic 3D optimization technique and associated software to recover configurations of density interfaces between the sedimentary load and the underlying basement. The density contrast variation within the volume of sedimentary pack follows exponential decay with depth. The forward modeling scheme used in Chapter-4 to calculate gravity anomalies is adopted here as a part the inversion module. Partial derivatives needed to frame normal equations are evaluated by numerical differentiation. The applicability

of the algorithm and software are validated with two gravity profiles, one theoretical and the other real. For real field example, the gravity data of the Almaz*á*n basin in NE Spain was analyzed and the inversion result was studied in the light of information derived from seismic data.

Chapter-6 presents the epitome of the entire work presented in this thesis and highlights the scope for future research.

CHAPTER
Two

2.75D - Automatic gravity modeling of sedimentary basins by means of growing polygonal source geometry with EDM*

2.1 General

Among the several applications of the gravity method, the unveiling of concealed basement configurations of sedimentary basins is significant. In general, sedimentary basins produce negative gravity anomalies because the density of in-filled sediments is less than that of the underlying basement. One can quantify these gravity anomalies for basement configurations in either profile mode or grid mode. In profile mode, a sedimentary basin shall be treated as either a 2D or 2.5D source, whereas in grid mode the source is considered as a 3D object. Under 2D approximation, the basin is treated as invariant (having uniform cross-section) throughout its strike, and the strike length is assumed as infinite. If the basin is considered as 2.5D source then its cross-section remains the same throughout its strike, and the strike length is treated as finite. In either case, the anomalies are digitized/scaled at appropriate intervals on a selective profile perpendicular to the strike of the basin. For 2.5D approximation, the profile along which the interpretation aimed is selected such that the profile more or less bisects the strike length of the basin. In the case of 2.75D, the profile runs across the strike, but at an offset. When interpretation is envisioned in 3D, anomalies are usually sampled at the nodes of a predefined regular grid.

Notwithstanding the fact that the density of sedimentary rocks seldom homogeneous, several automatic interpretation techniques treating the sediment density as uniform have been proposed since the times of Bott (1960) to analyze gravity anomalies resulted from sedimentary basins. Bott (1960) was the first researcher who had formulated an iterative method to analyze the gravity anomalies of sedimentary basins, wherein the sedimentary

---

* *Published in Journal of Geophysics and Engineering, 2017, 14, 303-3015.*

column above the interface was described with an ensemble of juxtaposed 2D strips. The thickness parameters of strips are adjusted till a satisfactory fit between the measured and theoretical gravity anomalies achieved. This approach of model simulation forms the basis for many new algorithms that were reported subsequently (See for e.g., Corbato, 1965; Murthy and Rao, 1989; Leão et al., 1996; Barbosa et al., 1997; Barbosa et al., 1999; Silva Dias, 2007). However, the necessary condition that the anomalies need to be specified at equi-interval on a profile becomes a constraint in the above techniques, including the Bott's (1960) method. The techniques proposed by Murthy et al. (1988, 1990) are free from such constraint, because their methods presume polygonal cross-section (Talwani et al, 1959) for basin geometry. All the above-mentioned methods are essentially 2D (the method of Murthy et al., 1990 also deals with 3D), and presume uniform density for the anomalous source, thence, they find restricted application in the interpretation of gravity anomalies of finite-strike basins in geological settings, where the density parameter warrants the use of non-uniform values with depth. Hence, it is necessary to develop new schemes of interpretation using appropriate depth-dependent density models apart from considering the sedimentary basin as finite-strike source (2.75D/2.5D).

2.2 Expression for gravity anomaly of a 2.75D sedimentary basin

The geometry of a typical finite-strike sedimentary basin in the Cartesian co-ordinate system is shown in Fig. 2.1, where the z-axis is considered positive downwards. The x-axis is placed transverse to the strike of the basin. Let $2L$ be the strike length of the basin, which is scaled along the y-axis. For the presumed geometry, the cross-sectional area of the basin remains the same throughout the strike in the xz plane. Furthermore, the color gradation within the basin from yellow to red signifies the decrease in sediment density contrast (absolute magnitude) with increasing depth. The gravity anomaly of such a model space at any observation, $P(0, s, 0)$, outside the source region, treated as 2.75D, can be expressed as (Chakravarthi et al., 2017a)

$$\Delta g_{2.75D}(0, s, 0) = G \int_v \frac{\Delta\rho(z) z \, dxdydz}{(x^2 + \overline{y - s}^2 + z^2)^{3/2}}. \tag{2.1}$$

Here, G is universal gravitational constant, (x, y, z) are coordinates of a volume element within the closed region, and $dxdydz$ represents the volume of an element. Also, s is the

offset distance of the profile, AB, scaled with reference to the origin $O(0,0,0)$ along the y-axis.

Substituting equation (1.12) for $\Delta\rho(z)$, equation (2.1) can be rewritten as

$$\Delta g_{2.75D}(0,s,0) = G\Delta\rho_0 \int_v \frac{ze^{-\lambda z}dxdydz}{(x^2+\overline{y-s}^2+z^2)^{3/2}}. \tag{2.2}$$

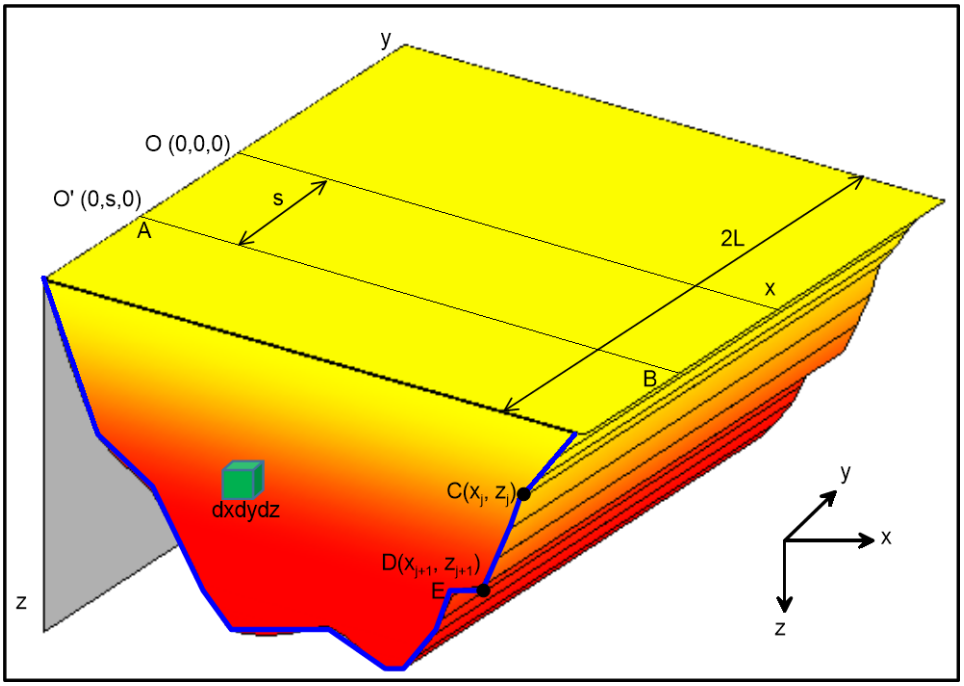


Fig. 2.1 Geometry of a 2.75D (strike-limited) sedimentary basin model (solid line in black), and its description by a multifaceted polygon source (solid line in blue). Other symbols used in the figure are described in the text (Chakravarthi et al., 2017a).

Integrating equation (2.2) with respect to y and substituting limits

$$\Delta g_{2.75D}(0,s,0) = G\Delta\rho_0 \int_S \frac{ze^{-\lambda z}}{(x^2+z^2)} \left[\frac{L-s}{\sqrt{(x^2+\overline{L-s}^2+z^2)}}\right.$$

$$\left. + \frac{L-s}{\sqrt{(x^2+\overline{L-s}^2+z^2)}}\right] dxdz. \tag{2.3}$$

Using Stokes' theorem equation (2.3) can be expressed as

$$\Delta g_{2.75D}(0,s,0) = G\Delta\rho_0 \oint_z e^{-\lambda z} \left[arctan \frac{x\,\overline{L-s}}{z\sqrt{(x^2+\overline{L-s}^2+z^2)}}\right.$$

$$+arctan\frac{x\,\overline{L+s}}{z\sqrt{(x^2+\overline{L+s}^2+z^2)}}\Bigg]dz. \tag{2.4}$$

Representing the shape of the basin by a strike-limited multifaceted polygon CDE…. (shown in Fig. 2.1 with a solid line in blue), x in equation (2.4) can be expressed in terms of z for the K^{th} side of the polygon bounded by vertices $C(x_k, z_k)$ and $D(x_{k+1}, z_{k+1})$ as

$$x = a + z\,coti. \tag{2.5}$$

Here,
$$a = x_k - z_k coti. \tag{2.6}$$

Also, the K^{th} side of the polygon (represented by the side, CD) subtends an angle i with the x-axis. Replacing x in equation (2.4) by equation (2.5), the gravity contribution of the K^{th} side, such as CD, can be obtained as (Chakravarthi et al., 2017a)

$$\Delta g_K(0,s,0) = G\Delta\rho_0 \oint_{z_k}^{z_{k+1}} e^{-\lambda z}\left[arctan\frac{(a+z\,coti)\,\overline{L-s}}{z\sqrt{((a+z\,coti)^2+\overline{L-s}^2+z^2)}}\right.$$
$$\left.+arctan\frac{(a+z\,coti)\,\overline{L+s}}{z\sqrt{((a+z\,coti)^2+\overline{L+s}^2+z^2)}}\right]dz. \tag{2.7}$$

The total anomalous field of the basin at $P(0,s,0)$, is then given by

$$\Delta g_{2.75D}(0,s,0) = \sum_{K=1}^{N} \Delta g_K(0,s,0), \tag{2.8}$$

where, N is the number of sides the polygon contains. Clearly, equation (2.7) cannot be solved analytically in the spatial domain. For this reason, the Simpson's rule had been used to obtain a numerical solution for equation (2.7) (Chakravarthi et al., 2017a). Equation (2.8) can also be used for forward modeling of a 2.5D source by setting s to zero.

2.2.1 Accuracy assessment of proposed forward modeling

To examine the correctness of the method, a theoretical model of a homogeneous strike-limited sedimentary basin is chosen (Fig. 2.2b), and its gravity response is calculated using equation (2.8). This response is then compared with the anomalies realized from analytical solution. The strike-length of the basin considered in this case is 40 km. Further, a total of 32 vertices defined with 32 pairs of arbitrarily chosen coordinates describe the geometry of

model space (Fig. 2.2b). Assuming -0.32 g/cm^3 as the density contrast between the sediments and the basement, theoretical gravity anomalies at 32 randomly distributed locations are generated on the profile between [0 km, 64 km] using equation (2.8) and shown in Fig. 2.2a (solid line in blue). It is to note that the profile chosen for anomaly calculation bisects the strike length of the structure and that the observer locations on the profile do not coincide with the x-coordinates of the model vertices. The anomaly calculation is repeated for the structure at the same observer locations using the analytic equation (Murthy, 1998) and shown in Fig. 2.2a as solid dots in red.

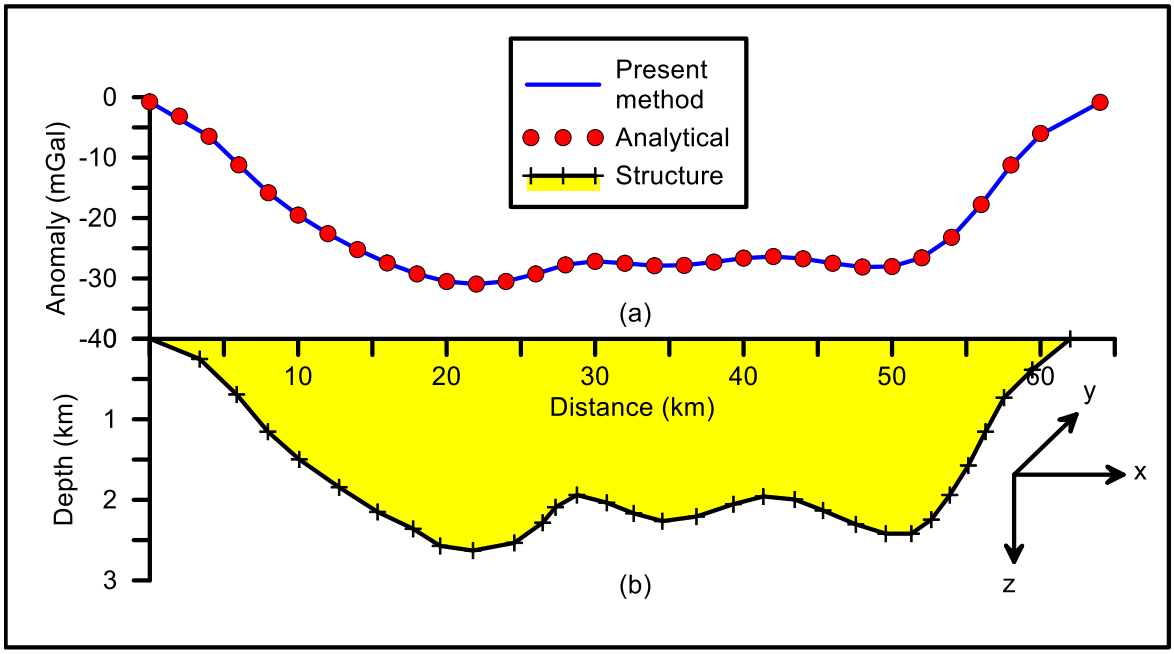


Fig. 2.2 (a) Theoretical gravity response of a 2.5D theoretical model by the present method at zero offset. Anomalies generated using analytical equation (Murthy, 1998) are also shown for comparison.

The residues between the magnitudes of two theoretical anomalies under consideration did not exceed ±1E-03 mGal, hence, the accuracy of equation (2.8) is verified.

2.2.2 Gravity signatures with variable density-depth models

As variable density models play important role in modeling studies, it is of paramount concern to examine the characteristics of gravity anomalies that are deemed to have been produced by sedimentary basins in which the density differs with depth in prescribed manner. Fig. 2.3a shows the gravity anomalies produced by a theoretical model of a 2.5D intracratonic sedimentary basin, whose floor is manifested by several inward dipping dip-slip faults with minor throws (Fig. 2.3b). The strike length of the basin is assumed as 40 km.

The simulated density models described in section 1.2 of Chapter-1 are used to compute the corresponding anomalous field of the basin (Fig. 2.3a) in the interval [0 km, 30 km] by setting the offset parameter, s, to zero in equation (2.7).

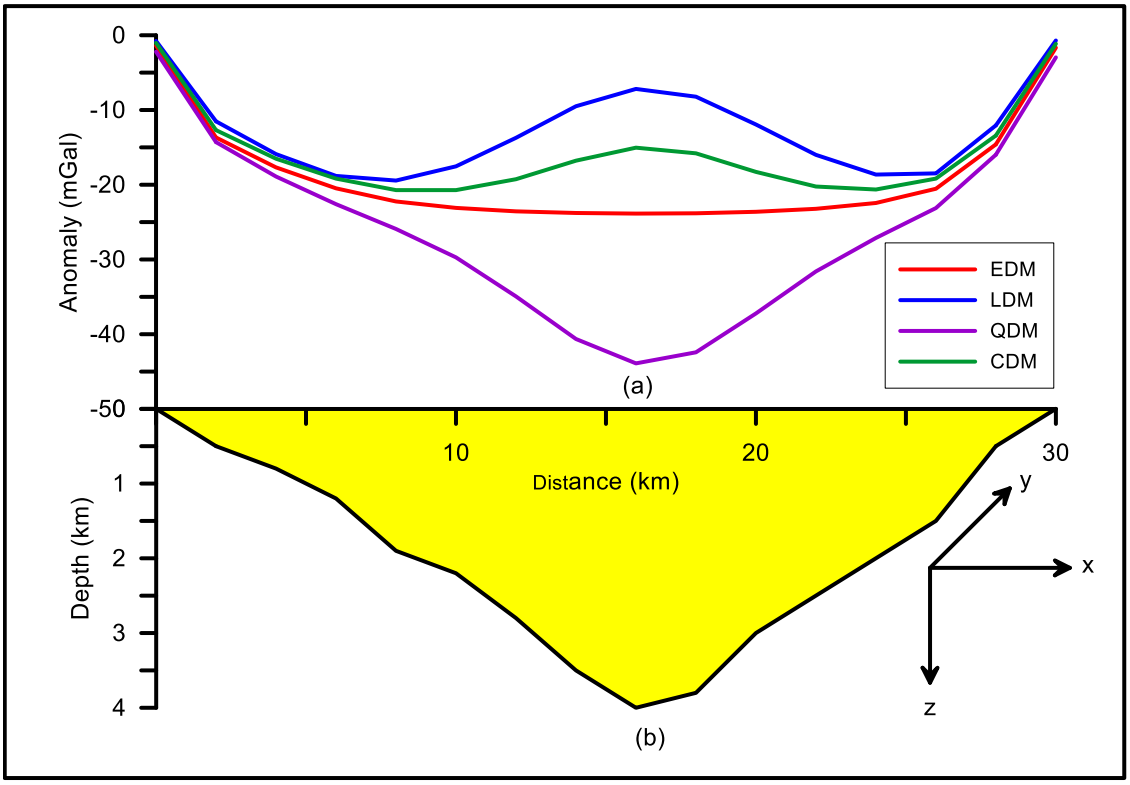


Fig. 2.3(a) Gravity response of a 2.5D sedimentary basin with prescribed EDM, LDM, QDM, and CDM (Table 1.1), (b) assumed depth structure.

It can be noticed from Fig. 2.3a that the anomalous field realized with EDM, and QDM shows increasing magnitudes (absolute) from either side of the margins of the basin towards the depocentre, whereas the anomalous field calculated with CDM and LDM shows unusual humps in the anomaly over the depocentral regions of the basin. Also, the anomalous field obtained with QDM shows exceedingly larger magnitude (absolute) than the anomalous field realized with EDM. The sediments beyond a depth of 2 km in case of QDM exhibits unusual increase (Fig. 1.1b) in density contrast (absolute), thereby produced a maximum anomaly (absolute) of 44 mGal over the depocentre against 23.8 mGal observed in case of EDM. In case of LDM, the positive gravity anomaly produced by the sediments beyond 1.5 km depth (because of positive density contrast) partly counteracts the negative anomalous field produced by sediments up to 1.5 km depth, thereby the overall magnitude of gravity anomaly got reduced (Fig. 2.3a). Similarly, the positive density contrast of sediments beyond 2 km depth, in case of CDM (Fig. 1.1b), is again responsible for producing unusual hump in the gravity anomaly over the depocentre (Fig. 2.3a). From above observations, it is clear

that use of LDM, QDM and CDM in the interpretation of anomalies leads to intricacies particularly when the said density models are built based on limited density-depth information. On the other hand, use of EDM in the interpretation ensures reliable results. It is to note that the anomalous field realized with EDM (Fig 2.3a) across the strike varies smoothly and does not succinctly reflect the structure features present in the interface.

2.2.3 Profile offset versus anomaly magnitude

To explore the effect of profile offset on the anomaly magnitude, two theoretical anomaly profiles were generated over the synthetic structure shown in Fig. 2.2b, one at the zero offset (profile bisecting the strike length) and the other at 18 km offset. In this case, the source x-coordinates coincide with observer locations on the profile. Further, the density contrast of sediments within the source follows exponential decease with depth (equation 1.12) defined with $\Delta\rho_0$= -0.322 g/cm^3 and λ = 0.31 km^{-1}. The anomalous field calculated in the range [0 km, 62 km], along each profile, at random intervals are shown in Fig. 2.4a.

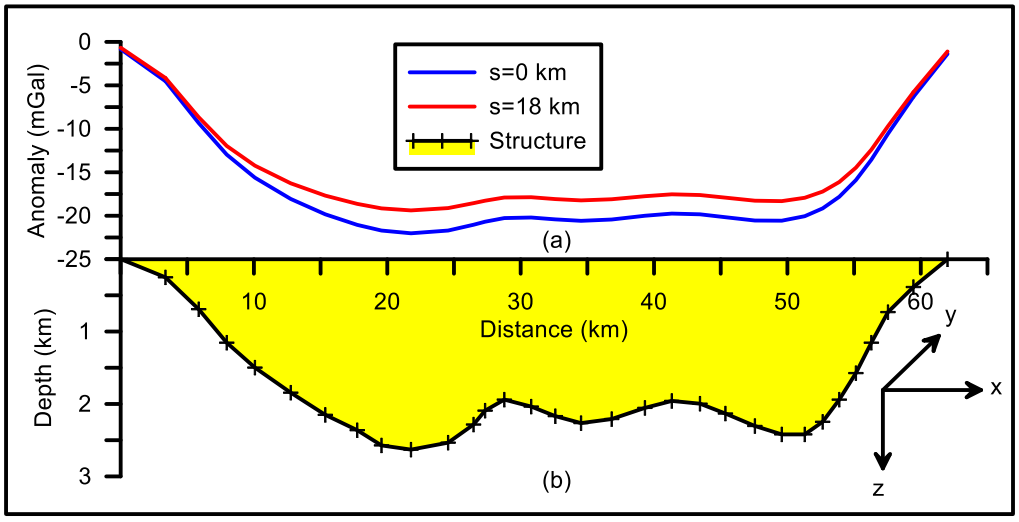


Fig. 2.4 (a) Gravity response of a sedimentary basin at two specific profile offsets, 0 km and 18 km with EDM, (b) basin geometry.

It is observed from Fig. 2.4a that the anomalous fields realized at 0 km and 18 km offsets differ in magnitudes despite the fact that the model space in both cases remain the same. In short, it is concluded that the magnitude of anomalous field (absolute) over a structure decreases with the increase in profile offset. Therefore, the offset parameter plays a decisive role in the interpretation of gravity anomalies. If the profile runs at an azimuth across the basin's strike, then the observer locations on the profile needs to be multiplied with $cos\alpha$, where α is the angle between the profile and x-axis.

2.3 Automatic modeling

To start with, the profile along which the interpretation envisioned is placed so that it runs across the strike and fully covers the lateral dimensions of the basin. The observer locations need not necessarily be placed/interpolated at equal intervals on the profile. Because the observer locations on the profile turn out to form the x-coordinates of the source (sedimentary basin) vertices, the problem becomes to estimate $(N_{obs} - 2)$ parametric values of z-coordinates from N_{obs} anomalies on the profile. A series of infinitely extending horizontal slabs, $(N_{obs} - 2)$ in number, whose top surfaces coincide with the observation plane are assumed responsible for generating the observed gravity anomalies on the profile. Under this approximation, it is assumed that the density contrast in each slab is also varying according to equation (1.12). The anomaly contribution, g_B, of one such slab can be calculated as (Cordell, 1973, Chakravarthi et al., 2016),

$$g_{B=}\frac{2\pi G\Delta\rho_0}{\lambda}\left(1-e^{-\lambda z_B}\right), \tag{2.9}$$

where, z_B is thickness of the slab.

Rearranging the terms, equation (2.9) can be expressed as

$$z_B = \frac{-1}{\lambda} log\left(1-\frac{\lambda g_B}{2\pi G\Delta\rho_0}\right). \tag{2.10}$$

Equation (2.10) forms the basis for initializing the model space. This would be realized by replacing g_B in equation (2.10) by the measured gravity anomaly, $\Delta g_{obs}(x_i)$, at each observation, x_i, i = 2, 3,…, $(N_{obs} - 1)$. Equation (2.8) calculates the model gravity response, $\Delta g_{2.75D}(x_i)$, of the initial model space at all observations x_i, i = 2, 3,…, N_{obs}. Because the initial depth parameters obtained from equation (2.10) are only tentative, the magnitude of model gravity response apparently deviates from the observed anomaly. In general, the misfit between these anomalous fields at the end of any iteration can be explained by a data misfit function (root mean square error) defined by Chakravarthi et al., (2017a) as

$$J_{rms} = \sqrt{\frac{\sum_{i=1}^{N_{obs}}[ERR(x_i)]^2}{N_{obs}}}, \tag{2.11}$$

where,

$$ERR(x_i) = \Delta g_{obs}(x_i) - \Delta g_{2.75D}(x_i). \quad (2.12)$$

The basin depths at all observations are adjusted, repeatedly, using the equation (Cordell and Henderson 1968; Blakely 1995)

$$z_{B_i}{}^{k+1} = \frac{\left[\ \Delta g_{obs}(x_i) * z_{B_i}{}^{k}\right]}{\Delta g^{k}_{2.75D}(x_i)}. \quad (2.13)$$

Here, k stands for iteration number. $z_{B_i}{}^{k+1}$ is modified depth estimate of the sedimentary basin at a location, x_i, on the profile. When compared to other methods (see for e.g. Cordell 1973; Granser, 1987; Feng et al., 2015), equation (2.13) has a conspicuous advantage to improvise the depth estimates in the sense that it is independent of the density contrast term. This formulation prevents the depth parameters to attain exorbitantly large values when the density contrast becomes too small (Mallesh et al., 2019).

Equation (2.8) calculates the gravity response of the improved model space at plurality of locations on the profile and a new data misfit, J_{rmsnw}, is obtained. If the value of current data misfit, J_{rmsnw}, is less than the previous one, J_{rms}, then the algorithm ascribes the value of J_{rmsnw} to J_{rms}, and the modified depth parameters $z_{B_i}{}^{k+1}$ to $z_{B_i}{}^{k}$. This process repeats in an iterative approach for a specific number of iterations, or up to the stage where the resulting data misfit equals to or falls below a predefined threshold, or when the resulting data misfit exceeds its preceding misfit.

2.3.1 POLYMODEXP

A software, POLYMODEXP, encrypted in Core Java has been developed to interpret the gravity anomalies of 2.75D (or 2.5D) sedimentary basins with EDM (Annexure 2-A). The methodology outlined in section 2.3 forms the basis for the software. This software works on Model-View-Controller (MVC) architecture as shown in Fig. 2.5. The advantage of the proposed MVC architecture is that it facilitates the user to customize the modules, if required. The architecture element 'Model' performs the tasks of model initialization, forward modeling, and execution of the business logic for model growth. The 'View' module enables the user to read input data to the software, and displays the interpretation

result in graphical as well as ASCII forms. The 'Controller' forwards needed actions to the modules of Model and View, whenever they called for.

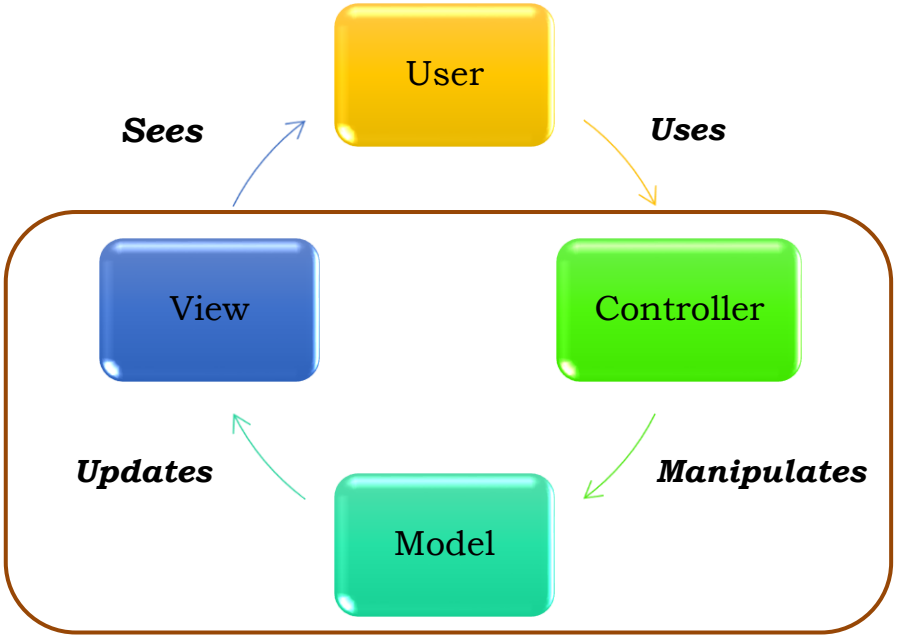


Fig. 2.5 Architecture and tasks of Model, View, and Controller (MVC)

Once the batch file of the software is invoked the view module appears on the monitor as shown in Fig. 2.6. The screen of the view module is arranged into three sub-layouts, namely, the input, graphical, and ASCII (Fig. 2.6).

Two options are available in the software to load input data. The user may enter the data in a formatted excel sheet, and then read it to the code by 'Load data' option button (Fig. 2.6). To avail this option, the user has to download the jar file jxl-2.6 (http://www. java2s.com/ Code/Jar/j/Downloadjxl26jar.htm) and save it in the executable folder of the software. Alternatively, the user may enter the data directly in the input layout by specifying the area name, profile name, number of observations on the profile, distance to observation (km), observed gravity anomaly (mGal), surface density contrast (g/cm^3), Lambda (km^{-1}), profile offset (km), half-strike length (km), minimum and maximum permissible depths to the interface (km), and number of iterations.

The code performs the interpretation within the specified convergence criteria by means of activating the action button 'Modeling'. Finally, the graphical layout displays the observed and modeled gravity anomalies in the top panel, model geometry in the bottom panel, changes in misfit norm and density-depth profile in the side panels respectively (Fig. 2.6).

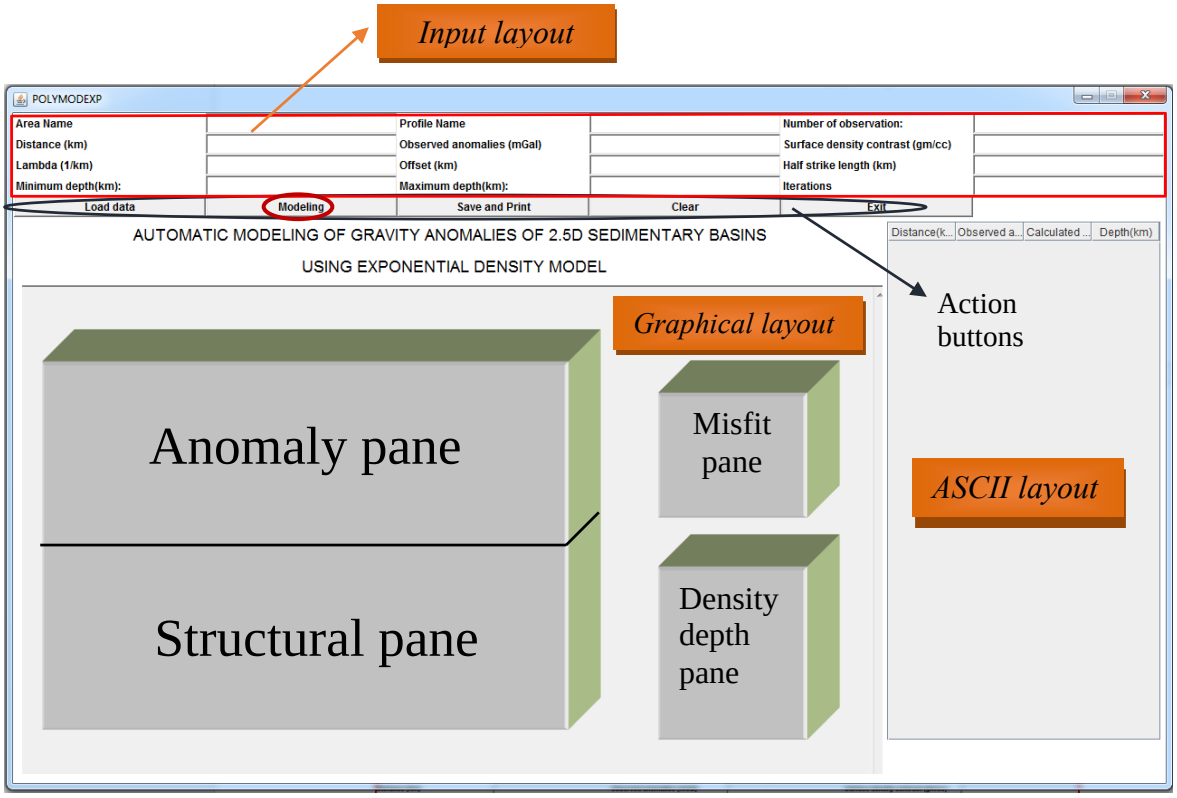


Fig. 2.6 View module of POLYMODEXP

The ASCII layout in the right hand side shows the output in a tabular form. The striking feature of the software is that it facilitates the user to visualize the animated versions in the changes of i) model space, and corresponding gravity response, ii) data misfit, and iii) density profile against the iteration number. Finally, the user may opt for 'Save and print' action button to save the output in an html format followed by printing of the result.

2.4 Applications

To establish the validity and applicability, the proposed method was initially applied to analyze the gravity anomalies attributable to a theoretical model followed by analyzing real world anomalies.

2.4.1 Theoretical Example

Gravity anomalies produced by a theoretical model (whose geometry is shown in Fig. 2.4b) at 18 km profile offset in the presence of random noise are shown Fig. 2.7a. The noise has zero mean and standard deviation of 0.1 mGal. This noisy data is considered as observed gravity anomalies (solid line in blue) to test the algorithm. The aim of present modeling is of two fold viz., i) to examine whether or not the proposed technique recovers the assumed

structure from the noisy anomalies, and ii) to quantify the effect of profile set on the interpretation. In either case, the density variation is well prescribed by EDM in the sense that the constants defining the density model are restrained from varying during the iterative process of modeling. In the first phase, profile offset was assigned to the optimum value of 18 km for which the technique took 15 iterations for a satisfactory convergence. The modeled gravity anomalies after the 15th iteration (shown as solid circles in red in Fig. 2.7a) almost coincide with the observed anomalies. The solid line in red in Fig. 2.8a shows errors between measured and theoretical gravity anomalies along the profile at the end of the concluding iteration.

The magnitudes of errors vary between -1.0E-04 to 5.0E-03 mGal, except at the 28th km where a maximum error of 0.011 mGal was found. Such magnitude of errors between the measured and theoretical gravity anomalies are tolerable because the anomalies used in the modeling are noisy. Fig. 2.8b depicts the overall change in data misfit (equation (2.11)) against the iteration number. The misfit for the initial model was 2.089 mGal, which had reduced sharply to 0.1301 mGal by the end of the 3rd iteration.

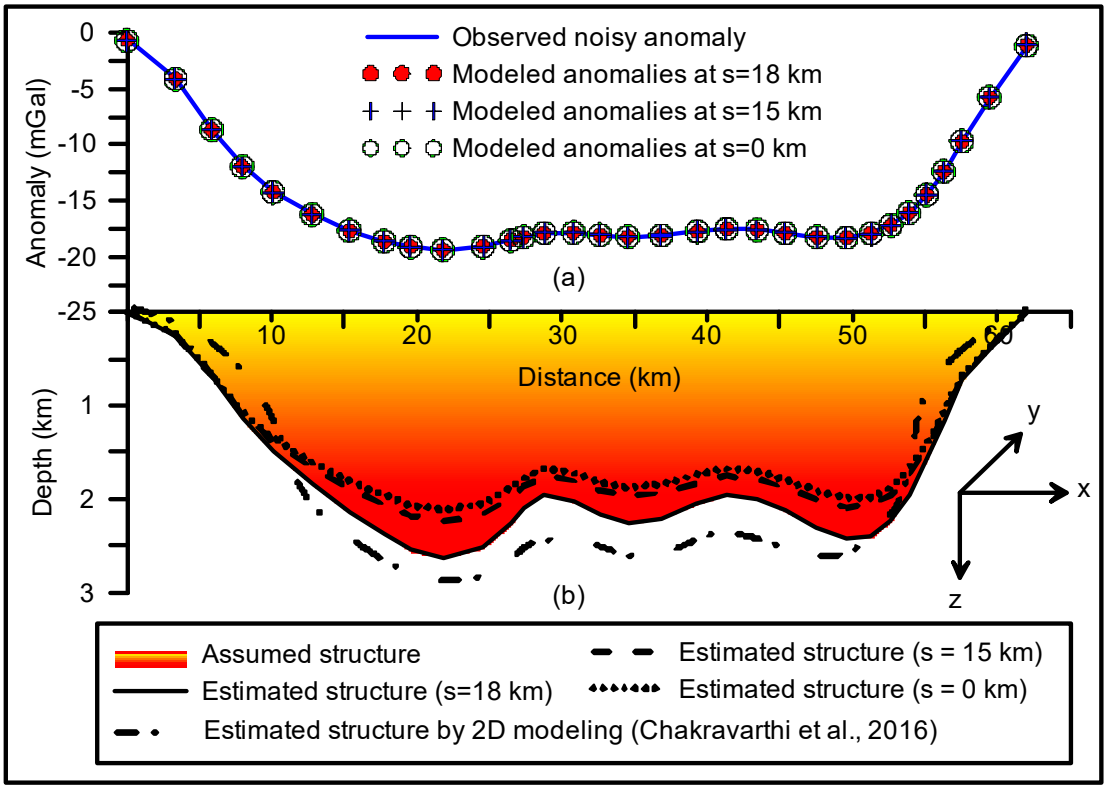


Fig. 2.7 (a) Observed gravity anomalies, and theoretical anomalies realized by automatic modeling (at different offsets), (b) presumed, and estimated depth models at different profile offsets, synthetic example (Chakravarthi et al., 2017a). Derived structure from 2D modeling of gravity anomalies is also shown for comparison.

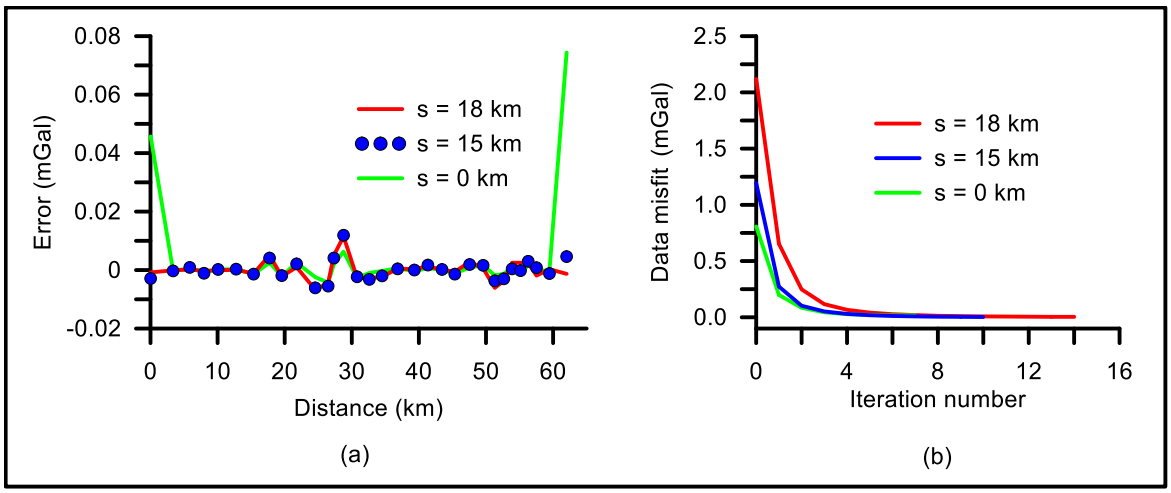


Fig. 2.8 (a) Error between observed and theoretical gravity responses, (b) changes in data misfits with iteration number for different profile offsets used in modeling, theoretical example.

Thereafter, the decay in misfit was somewhat slow with the increase in iteration number (Fig. 2.8b). Finally, at the end of the 15th iteration the data misfit had attained a value of 0.0498 mGal, which was less than the predefined threshold (0.05 mGal). The algorithm was terminated at this stage and the estimated depth structure corresponding to this minimum misfit is shown in Fig. 2.7b. It is noticed from Fig. 2.7b that, by and large, the recovered structure closely resembles the assumed one although some insignificant deviations are inevitable. The differences between the presumed and inferred depths are within ±1% error (Fig. 2.9), which are acceptable.

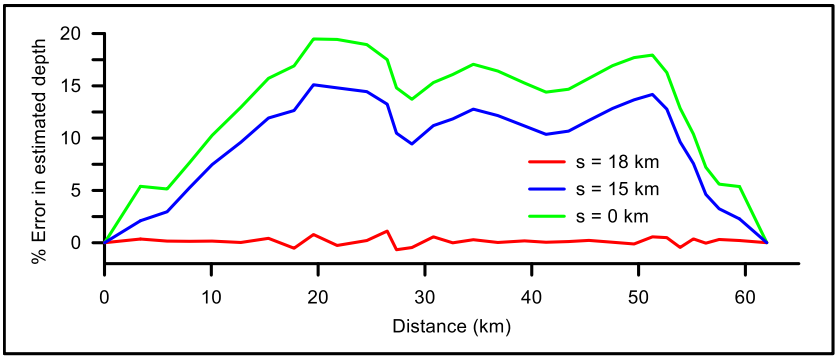


Fig. 2.9 Errors (%) between assumed and estimated depths for different profile offsets used in modeling, synthetic example.

In the second phase, the modeling process was repeated on the noisy anomalies shown in Fig. 2.7a by letting profile offset in each case to 15 km and 0 km, respectively. The code had performed 12 iterations for 15 km profile offset, and mere 7 iterations in case of 0 km offset. The data misfits subsequent to the respective concluding iterations have attained higher

magnitudes than their preceding counterparts, thereby forced the algorithm for its termination. It is to note from Fig. 2.8a that the magnitudes of errors between measured and theoretical gravity anomalies observed in either case remain more or less the same over a larger part of the profile, except at the first and last observations (Fig. 2.8a). This observation reveals that the modeled gravity anomalies in either case have explained the observed anomalies equally well irrespective of the profile offset being chosen in the modeling. Furthermore, the decay of misfit norm in either case is almost commensurate to the one observed with 18 km offset despite the initial misfits differ from each other (Fig. 2.8b). The estimated depth structures obtained by setting s = 15 km offset and s = 0 km offset are also shown in Fig. 2.7b. The corresponding errors (%) between the interpreted and actual depths of the structure are shown in Fig. 2.9. Figs. 2.7 and 2.9 reveal that the choice of 15 km profile-offset in the interpretation leads to underestimate the structure by 12-14%, whereas it was more than 16-18% in case of 0 km offset. Even the choice of optimum profile-offset (s=18 km) in the interpretation also leads to a structure that has minor deviations from the assumed one (Figs. 2.7b and 2.9). However, the deviations observed in the estimated structure with 18 km offset are considered as insignificant keeping in view of the fact that the anomalies used in the interpretation are associated with considerable noise.

The observed anomaly shown in Fig. 2.7a was also interpreted by a 2D modeling technique (Chakravarthi et al., 2016) to explore its effect on the estimated model. In this case, the modeled anomalies after the 31st iteration closely resembles the observed anomalies. However, the depths of the interface were underestimated on either side of the basin over its shoulders, whereas at depocentral regions they are overestimated (Fig. 2.7b).

2.4.2 Field example

A total of three field gravity anomaly profiles, two from the Gediz graben and one from the Büyük Menderes graben, in western Turkey are interpreted by the proposed modeling. The interpreted results are discussed in the light of available/reported information. The two grabens are formed in a tectonically active extension zone boarded by east-west trending normal fault systems. The Büyük Menderes graben joins the slightly convex shaped Gediz graben towards the east (inset of Fig. 2.10). Each graben has witnessed widespread sediment deposition over the metamorphic basement. Major tectonic elements of the grabens together with the positions of the gravity profiles among which interpretations performed are shown in Figs. 2.10 and 2.11. The noteworthy features of the grabens are that the main fault of the

Gediz graben is located on the south (Fig. 2.10), whereas in the Büyük Menderes graben the master fault is towards the north (Fig. 2.11).

Albeit the grabens under study are of finite strike lengths, about 150 km each, Paton (1992) had opted 2D methodology to analyze the gravity data pertaining to the grabens considering uniform density. Later, Sari and Şalk (2002) had demonstrated that the density of sedimentary rocks within these grabens vary non-uniformly with increasing depth. They used the hyperbolic density model (HDM) to describe the density variation as a function of depth in the grabens and adopted them to analyze the gravity profiles; two in the Gediz graben (locations are shown in Fig. 2.10), and one in the Büyük Menderes graben (location is shown in Fig. 2.11). It is to note that the approach adopted by Sari and Şalk (2002) to analyze the anomalies is again 2D.

As opined by Chakravarthi and Sundararajan (2007a), these grabens are treated as 2.75D sources rather than 2D, and accordingly the anomaly profiles over the grabens (shown in Figs. 2.10 and 2.11) are interpreted. Each graben was simulated with a polygonal shape, and appropriate profile offsets are assigned in modeling. Further, to accommodate non-uniform density variation in modeling, equation (1.12) was fitted to the derived density profiles of respective grabens (Sari and Şalk, 2002) and constants of EDM were worked out and shown in Table 2.1. Fig. 2.12 shows graphical representations of the derived density models.

Table 2.1

Derived density models of the Gediz and Büyük Menderes grabens, western Turkey (after Chakravarthi et al., 2017a)

Name of the graben/profile no.	*HDM (Sari and Şalk, 2002)*		*(EDM) (Chakravarthi et al., 2017a)*	
	$\Delta\rho_0$ (gm/cm^3)	β (km)	$\Delta\rho_0$ (gm/cm^3)	λ (km^{-1})
Gediz/2	-1.407	0.859	-1.182	1.019
Gediz/3	-1.407	0.620	-1.320	1.6708
Büyük Menderes/XY	-0.98	2.597	-0.798	0.3808

The measured gravity anomalies along profiles 2 and 3 of the Gediz graben are shown in Figs. 2.14a and 2.14c and that of XY profile in the Büyük Menderes graben in Fig. 2.14e, respectively. When the anomalies are modeled using the present technique, acceptable convergence between the observed and theoretical gravity anomalies has been realized after the end of the 23rd iteration for profile 2, and 26th iteration for profiles 3 and XY, respectively.

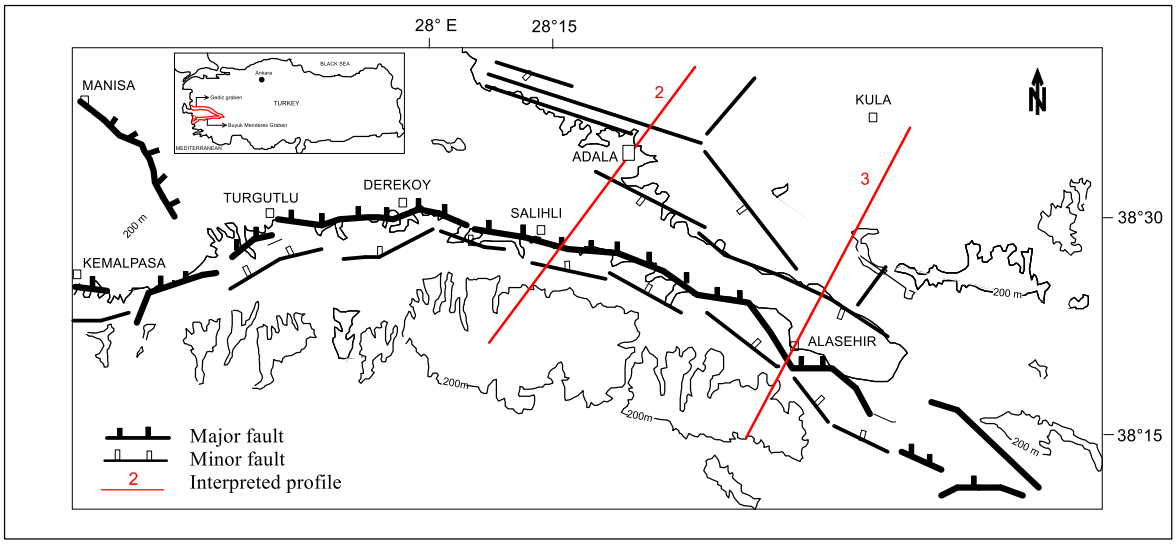


Fig. 2.10 Tectonic elements of the Gediz graben, western Turkey. Thick and thin lines in black indicate major and minor faults, respectively. Solid and open ticks on the lines indicate downthrown sides Solid lines in red are the gravity profiles among which interpretations have been carried out by the present method. Inset figure shows disposition of the Gediz and Büyük Menderes grabens (after Paton, 1992; Sari and Salk, 2002; Chakravarthi et al., 2017a).

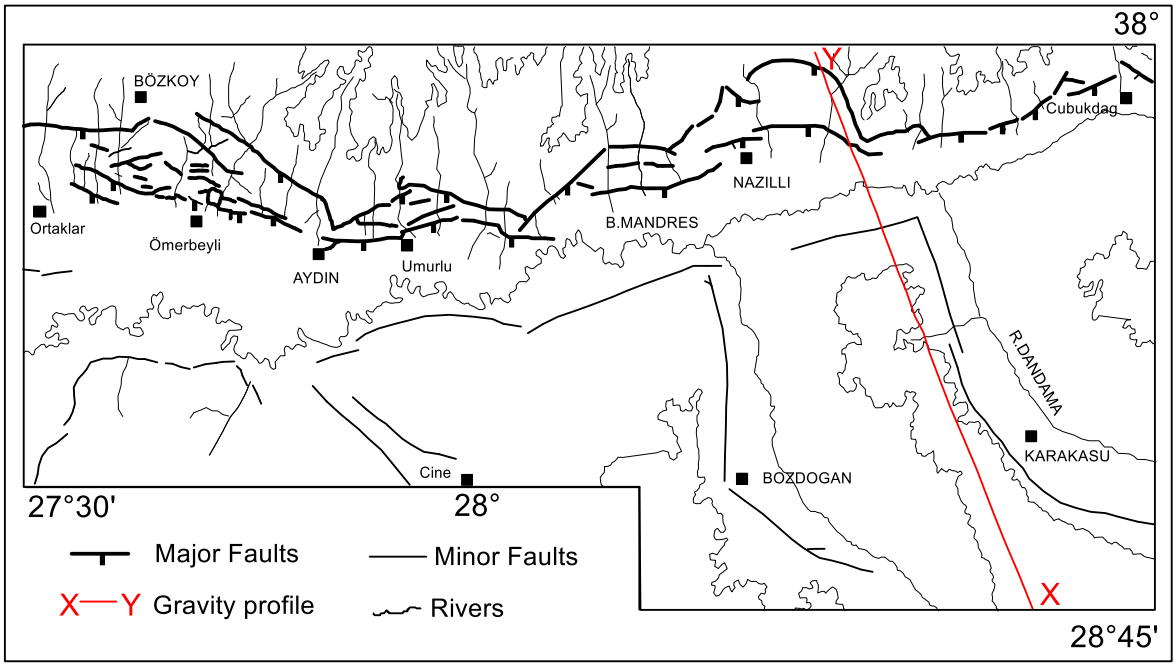


Fig. 2.11 Structural features of the Büyük Menderes graben, western Turkey. Thick and thin lines in black indicate major and minor faults, respectively. Solid ticks on the thick line indicate downthrown sides (after Paton, 1992; Sari and Salk, 2002; Chakravarthi et al., 2017a). Solid line in red is the gravity anomaly profile along which the interpretation is performed using the present method.

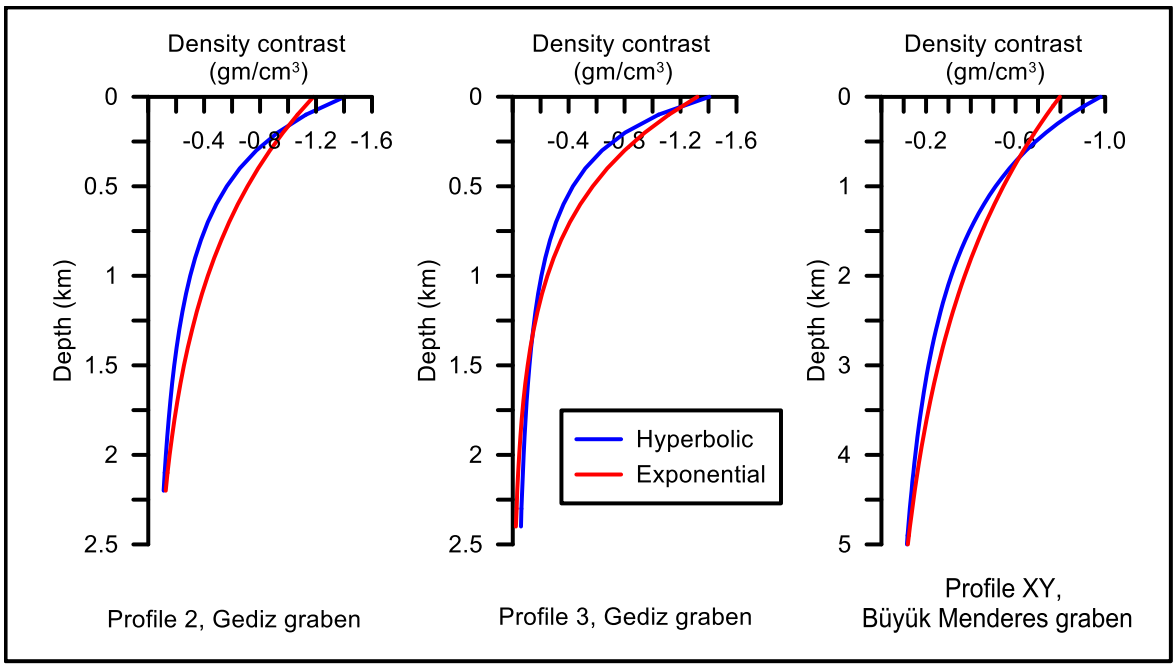


Fig. 2.12 Estimated EDMs for the Gediz and the Büyük Menderes grabens, western Turkey (Chakravarthi et al., 2017a)

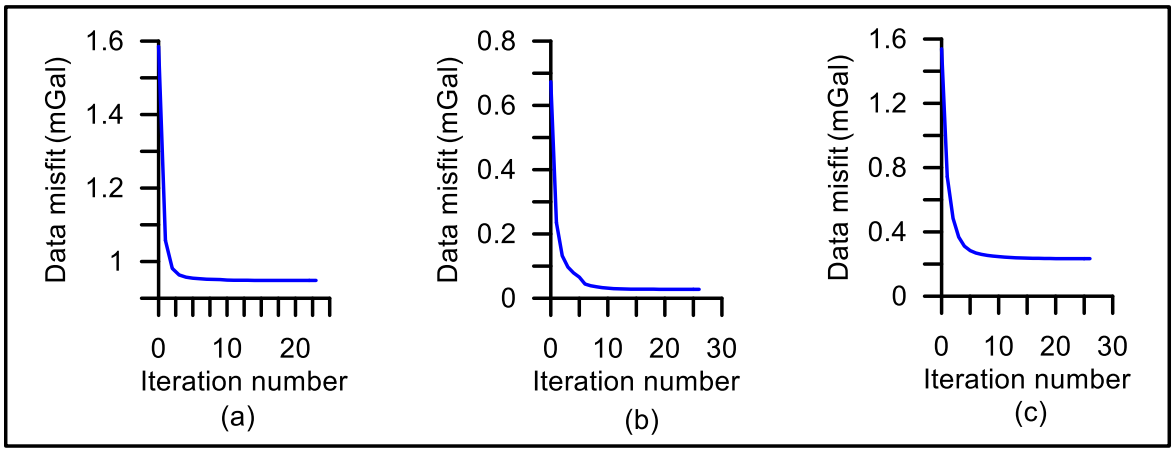


Fig. 2.13 Variations in data misfits against iteration (a) profile 2, (b) profile 3, and (c) profile XY, field example.

For profile 2, the misfit norm (equation 2.11) after the 23rd iteration (0.9485 mGal) was more than the previous misfit (0.94844 mGal), hence the algorithm got terminated. Similarly, for profile 3 and profile XY the corresponding misfits after the 26th iteration (0.02794 mGal for profile 3 and 0.23317 for profile XY) were noticed more than their previous misfits (0.02792 mGal for profile 3 and 0.23314 mGal for profile XY), thereby forced the algorithm for its termination. The overall variations in data misfits against the iteration numbers for three profiles under discussion are shown in Fig. 2.13. It is noticed from Fig. 2.13 that data misfits in all cases have decreased exponentially with iteration numbers.

The obtained theoretical gravity anomalies after the concluding iterations for profiles 2 and 3 of the Gediz graben and profile XY of the Büyük Menderes graben have shown satisfactory fit against the corresponding measured anomalies (Figs. 2.14a, 2.14c, and 2.14e). The estimated structures associated with minimum misfit norms are shown in Figs. 2.14b, 2.14d and 2.14f. The interpreted models by Sari and Şalk (2002) among the same profiles are also shown in respective figures for comparison. On the whole, the models of grabens deciphered from the present modeling and that of Sari and Şalk (2002) share some common structural features (Fig. 2.14).

However, a few minor deviations exist between the said inferred models in all the profiles. For e.g., in all cases the present modeling has yielded relatively deeper basement for the grabens over their shoulders (except on the NW part of the profile XY), whereas the derived models by Sari and Şalk (2002) do not. Also, the present technique places the basement at relatively shallower depths over the depocentres when compared to the models of Sari and Şalk (2002). The interpretations of two grabens from present method appears to be more sensible than the corresponding ones of Sari and Şalk (2002) for the obvious reason that 2D interpretation of gravity anomalies of a finite-strike source always result in underestimating the structure over the shoulders and overestimating the structure at depocentral regions.

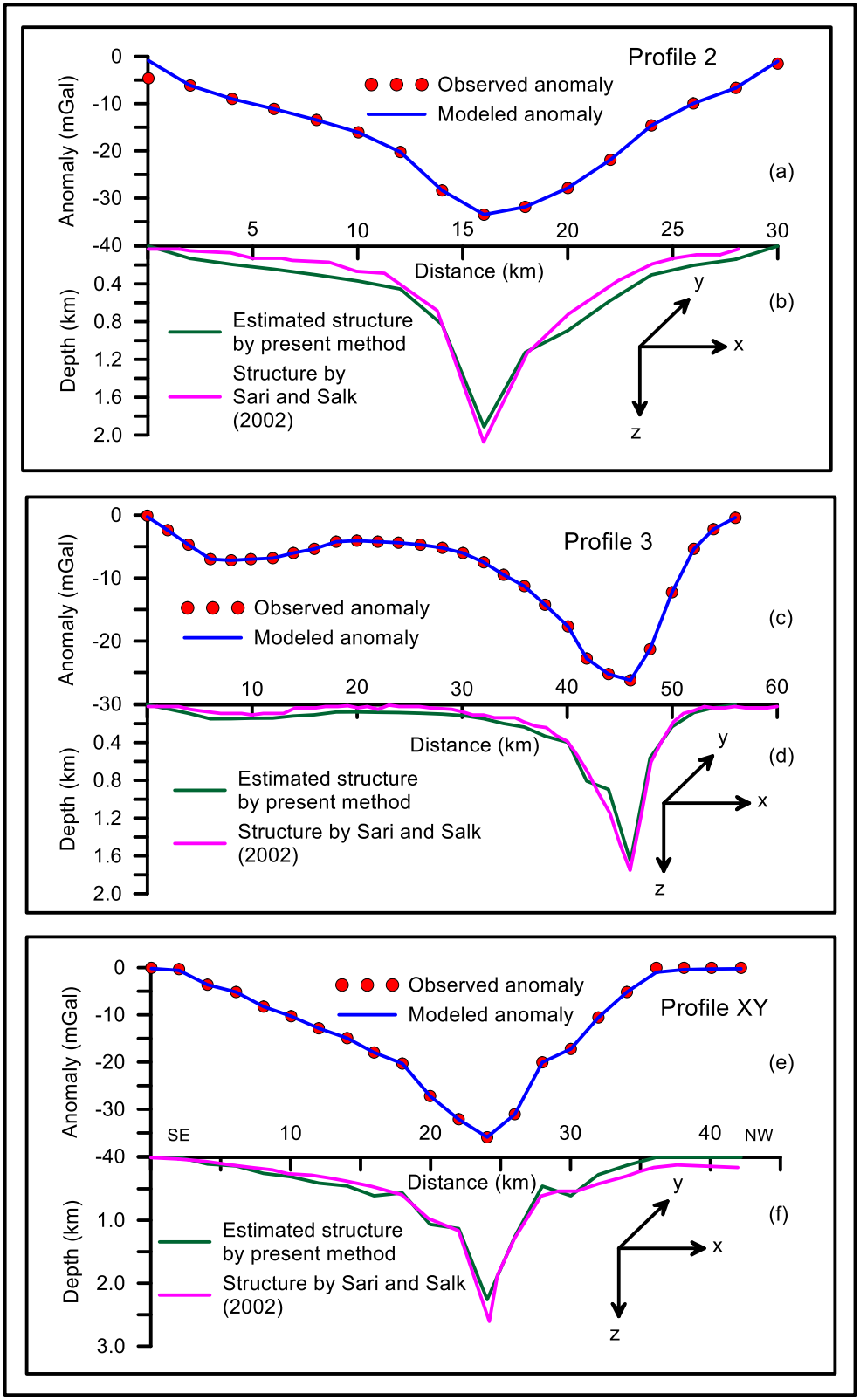


Fig. 2.14 Measured, theoretical gravity anomalies (a,c,e), and estimated structures from present modeling (b, d, f) - Profile 2 and Profile 3 of the Gediz graben and profile XY of the Büyük Menderes graben, western Turkey. Interpreted models by Sari and Şalk (2002) are also shown in respective depth panels for comparison (Chakravarthi et al., 2017a).

2.5 Results and discussion

i) A new 2.75D automatic modeling technique and related GUI software, coded in Core Java, are presented to quantify gravity anomalies generated by finite-strike sedimentary basins among which the density contrast variation follows prescribed exponential law.

ii) In contrast to approximating the sedimentary basin fill by a collage of vertical prisms, as in the case of many existing methods, this technique approximates the density interface by a finite-strike polygon geometry.

iii) The ordinates of several vertices of the polygon form the unknown parameters that need to be solved from a given set of gravity anomalies.

iv) As the exponential density model precludes one from deriving analytical equations for gravity anomalies in the space domain, the proposed method computes the model gravity response by both analytic and numeric methods.

v) The correctness of the proposed forward modeling is established over a homogenous synthetic model by means of comparing the model gravity response obtained from the proposed method with the anomalies computed analytically.

vi) It is shown that the anomalous field produced by a finite-strike source at different profile-offsets is different.

vii) The applicability of the present automatic modeling technique is demonstrated on a synthetic model gravity response followed by two field gravity anomalies.

viii) In the case of theoretical example, the modeling technique was successful in recovering the assumed structure of a strike-limited sedimentary basin though the structure anomaly contains considerable pseudorandom noise.

ix) The effect profile-offset in automatic modeling of gravity anomalies is discussed in length on the same synthetic model. It was shown that inappropriate choice of profile offset in modeling would result in yielding a structure that is grossly underestimated.

x) The real field application of the method over three gravity anomaly profiles, two from the Gediz graben and one from the Büyük Menderes graben, has yielded structure models that are consistent with the reported information. The deviations in the structural models exist between the present ones and the ones reported by Sari and Şalk (2002) are discussed.

xi) One of the major advantages of the present method is that the technique is fairly applicable even when the profile of interpretation fails to bisect the strike length of

a sedimentary basin, and also the anomalies need not necessarily specified at equal interval.

xii) The software, POLYMODEXP, is user friendly, and easy to operate.

CHAPTER
Three

2.75D - Automatic optimization of sedimentary basin gravity anomalies by polygonal source geometry with EDM*

3.1 General

Inversion techniques are very popular in optimizing geophysical anomalies to decipher the optimum dimensions of concealed geologic sources/physical properties. Though the aim of both automatic modeling and inversion techniques is more or less the same, the way in which they handle the data for analysis differs from each other. In both approaches, an appropriate model geometry is assigned to the causative source based on the nature and characteristics of the observed anomaly. A set of approximate parameters are assumed to define the model space and a suitable forward modeling scheme computes its theoretical response. Subsequently, misfit between the measured and theoretical anomalies shall be assessed and model space is improved accordingly in successive manner based on some fitting criteria. Automatic modeling schemes mostly use forward difference approximation, whereas inversion techniques construct and solve the system of normal equations for estimating the improvements in unknowns. Interpretations realized from automatic modeling are likely to be affected in case residual anomalies used in the analysis contain regional component. In such cases, inversion techniques are inevitable to use for handling such data sets for dependable interpretations.

It is true that strike lengths of sedimentary basins are mostly finite on the continental regions (Peirce and Lipkov, 1988). Mickus and Peeples (1992) had developed an inversion technique to find the thickness parameters of 2.5D basins from the measured gravity and magnetic anomalies. The theory of Backus and Gilbert (1967, 1968, and 1970) is the basis for this technique. Though this technique is popular in theoretical point of view, its application to

---

** Published in Journal of Geophysics and Engineering, 2017, 14, 303-3015.*

analyze real field anomalies is somewhat restricted for the obvious reason that the density of sedimentary rocks never stands uniform. Murthy and Rao (1979) and Rao and Murthy (1989) have developed schemes to compute the gravity contributions of polygonal cross-sections in the spatial domain, wherein the exponential density variation was simulated by several piece-wise linear density functions. Undoubtedly, these schemes are very efficient for anomaly calculation, but, they consume large computational time (Rao et al., 1994; Chakravarthi et al., 2017a).

Because no closed-form equations could exist or derivable in the spatial domain for anomaly calculation with EDM, several researchers have inclined towards the frequency domain to derive anomaly equations for the gravity anomalies. To name a few, Cordell (1973) had proposed a method combining both the gravity field and its gradient to decipher sedimentary basin structure from the anomalies using EDM. Making use of the Tylor series approximation and the application of FFT, Granser (1987) had developed a method to realize forward modeling of sedimentary basins using EDM. Chai and Hinze (1988) calculated the anomalies of prism-shaped bodies in the spectral domain, and for further analysis these anomalies were transformed back to the space domain by a shift-sampling method. Rao and Rao (1999) also had followed almost a similar approach to compute the anomalies in the frequency domain but they opted the Filon's (1928) method for transformation of anomalies. A more generalized approach to compute gravity anomalies of basins that are bounded by non-smoothing surfaces had been proposed by Chappel and Kusznir (2008) in the wave number domain. Except the methods of Granser (1987), Chai and Hinze (1988), rest of the methods are essentially 2D. However, popping up of truncation errors during the transformation of anomalies from wave-number to space domain is a major snag associated with all the above-mentioned methods (Chakravarthi and Sundararajan, 2007b; Chakravarthi et al., 2016). Rao et al. (1990b) had used a quadratic density function (QDF) in a 2.5D modeling technique to analyze the anomalies resulted from sedimentary basins. Though this technique considers finite strike-length for sedimentary basins, non-consideration of profile offsets in forward modeling is a major constraint. Besides, the drawbacks associated with QDF are overt as detailed in Chapter-1.

In the past, Chakravarthi et al. (2013a) have developed an algorithm and a software (with GUI compatibility) in the spatial domain to quantify anomalous gravity fields of 2.75D sedimentary basins using EDM. Although this technique has an affinity to the Bott's (1960)

method in its application, it differs from the later in the sense that the prisms used in model construction are of variable but finite strike lengths. Profile offsets measured from the centre of each prism were also been taken care of by the method of Chakravarthi et al. (2013a). Though efficient, this technique can't be applied if the anomalies sought for the interpretation are available at random interval. Therefore, it is necessary to develop an efficient inversion technique that is capable of analysing the anomalies available at random intervals in the presence of regional component. This chapter briefs the development of an optimization technique and relevant software to interpret the anomalies generated by a finite strike sedimentary basins using prescribed EDM. To show its applicability, the technique has been applied to recover the basement geometry of a synthetic model from its noisy anomalies both with and without regional component. Real field application of the technique is demonstrated on the anomalies of the Gediz and Büyük Menderes grabens of western Turkey.

3.2 Expression for Bouguer gravity anomaly of a 2.75D source

Upon simulating the geometry of a sedimentary basin by finite-strike polygonal source as indicated in Fig. 2.1, the gravity anomaly of the structure at an observation can be expressed in the presence of regional component as (Chakravarthi et al., 2017a)

$$\Delta g_T(0,s,0) = \Delta g_{2.75D}(0,s,0) + \sum_{i=0}^{N1} f_i x^i. \qquad (3.1)$$

Here, the polynomial term $\sum_{i=0}^{N1} f_i x^i$ represents regional gravity anomaly at any observation, x, on the profile and $N1$ stands for the prescribed degree of defined polynomial. It is presumed that regional field always has broad wavelength characteristics compared to the structure anomaly, and that these two anomalies have non-overlapping frequencies. In principle, any degree can be assigned to the polynomial to simulate the regional field along a selected profile; but, in this case 1st order is chosen for the obvious reason that the total number of parameters to be solved from gravity inversion should never be exceed the number of observations. Here, the total number of depth parameters to be solved from the measured anomalous field is, $(N_{obs} - 2)$; and by approximating the regional field by a linear equation as a function of x, additional 2 parameters pertaining to the coefficients of the polynomial needs to be solved. Such a formulation makes the solvable parameters to $(N_{obs} - 2 + 2)$ in number, which equals the number of anomalies, N_{obs}. In case, a higher degree polynomial is the option to be chosen to simulate regional field then the number of

depth parameters to be estimated from inversion must be cut down accordingly. If regional background is described by a linear equation, then equation (3.1) can be written as (Chakravarthi et al., 2017a)

$$\Delta g_T(0,s,0) = \Delta g_{2.75D}(0,s,0) + \sum_{i=0}^{1} f_i x^i. \qquad (3.2)$$

Here, $\Delta g_{2.75D}(0,s,0)$, is given by equation (2.8). It is to note that equation (3.1) can accommodate either a constant regional throughout the length of the profile or regional field that follows a linear trend.

3.3 Optimization of gravity anomalies

Optimization of gravity anomalies of sedimentary basins means fitting the anomaly equation (3.2) to the measured gravity anomalies in a least square approach for estimating the unknown parameters within the predefined limits.

Having described the cross-section of a sedimentary basin by a finite-strike polygon, the present inversion starts by initializing the model space based on the observer locations on the profile and depth parameters estimated from equation (2.10). Theoretical anomalies of this initial model are calculated using equation (3.2), where coefficients of the polynomial, f_0 and f_1 are initially set to zero. These theoretical anomalies, $\Delta g_T(x_i)$ realized from equation (3.2) apparently deviate from the measured anomalies, $\Delta g_{obs}(x_i)$, because the initial depths used in forward modeling are approximate. The difference in the two anomalies at an observation, x_i, on the profile can be vented in the form of a truncated Tylor series expansion as (Chakravarthi et al., 2017a)

$$\Delta g_{obs}(x_i) - \Delta g_T(x_i) = \sum_{j=2}^{N-1} \frac{\partial \Delta g(x_i)}{\partial z_j} dz_j + df_0 x_i + df_1. \qquad (3.3)$$

Here, dz_j represents the improvements in depth ordinates of the polygon, and df_0 and df_1 represent improvements in the polynomial coefficients, respectively. Linear equations akin to equation (3.3) are framed for all observations on the profile, and N normal equations are constructed and solved for the N unknown parameters by minimizing the data misfit given by equation (2.11) using the Marquardt's (1970) algorithm.

The normal equations to be solved for the improvements in unknown parameters can be expressed in a matrix form as (Chakravarthi et al., 2017a)

$$(A + \delta I)X = B. \tag{3.4}$$

Here, A is NxN matrix containing partial derivatives of the anomaly. The elements, $A_{nj'}$, of the matrix are given by

$$\sum_{n=1}^{N}\sum_{m=1}^{N}\frac{\partial\Delta g(x_m)}{\partial a_{j'}}\frac{\partial\Delta g(x_m)}{\partial a_n}, j' = 1, 2, \dots, N. \tag{3.5}$$

The parameter matrix X is defined as

$$X = da_n, \tag{3.6}$$

and the matrix, B, containing error term of the anomaly is given by

$$B = \sum_{m=1}^{N}[\Delta g_{obs}(x_m) - \Delta g_T(x_m)]\frac{\partial\Delta g(x_m)}{\partial a_{j'}}, j' = 1, 2, \dots, N. \tag{3.7}$$

Also, $a_n = z_n$, n = 1, 2,…., $N-2$, $a_{N-1} = f_0$, and $a_N = f_1$.

In equation (3.4), δ represents the damping factor whose value is initially set to 0.5. Subsequently, its magnitude is controlled by the algorithm in automatic sense depending upon whether the data misfit at the end of a specific iteration is more or less when compared to the previous misfit. Its value shall be decreased sequentially by a factor of two as long as the resulting data misfit portrays continuous decay with iteration number. In case the data misfit at a specific iteration attains a higher magnitude than its preceding value, then the existing damping factor is doubled and the system of normal equations given by equation (3.4) is solved for the improvements in unknown parameters. This exercise is repeated several times within the inner loop of corresponding iteration till the resulting misfit attains a magnitude lesser than the misfit of preceding iteration. Detailed description of the application of Marquardt's (1970) algorithm was given by Murthy (1988), Chakravarthi (2003), and Chakravarthi et al. (2006b). Further, I in equation (3.4) represents a diagonal matrix whose elements are given by the diagonal elements of matrix, A. Because the exercise of forward modeling (equation (3.2)) involves both analytic and numeric approaches, the partial derivatives required in equations (3.5) and (3.7) are computed by a numerical method (Chakravarthi et al, 2013a). Solution of equation (3.4) yields the improvements, da_n, in the

unknown parameters, a_n. The model space and coefficients of the polynomial are updated using the estimated parameter improvements, da_n. The algorithm repeats the procedure till the data misfit (equation 2.11) reaches a value less than or equal to the predefined threshold. The inversion process also terminates upon the completion of specific number of iterations or when the damping factor, δ, attains to an unusually large magnitude (Murthy, 1988; Chakravarthi, 2003).

3.3.1 POLYINVEXP

Based on the methodology described in section 3.3, a software, POLYINVEXP, coded in Core Java has been developed to interpret the gravity anomalies of finite-strike sedimentary basins using exponential density model (Annexure 3-A). Fig. 2.5 briefs the overall architecture of the software and the roles of different modules.

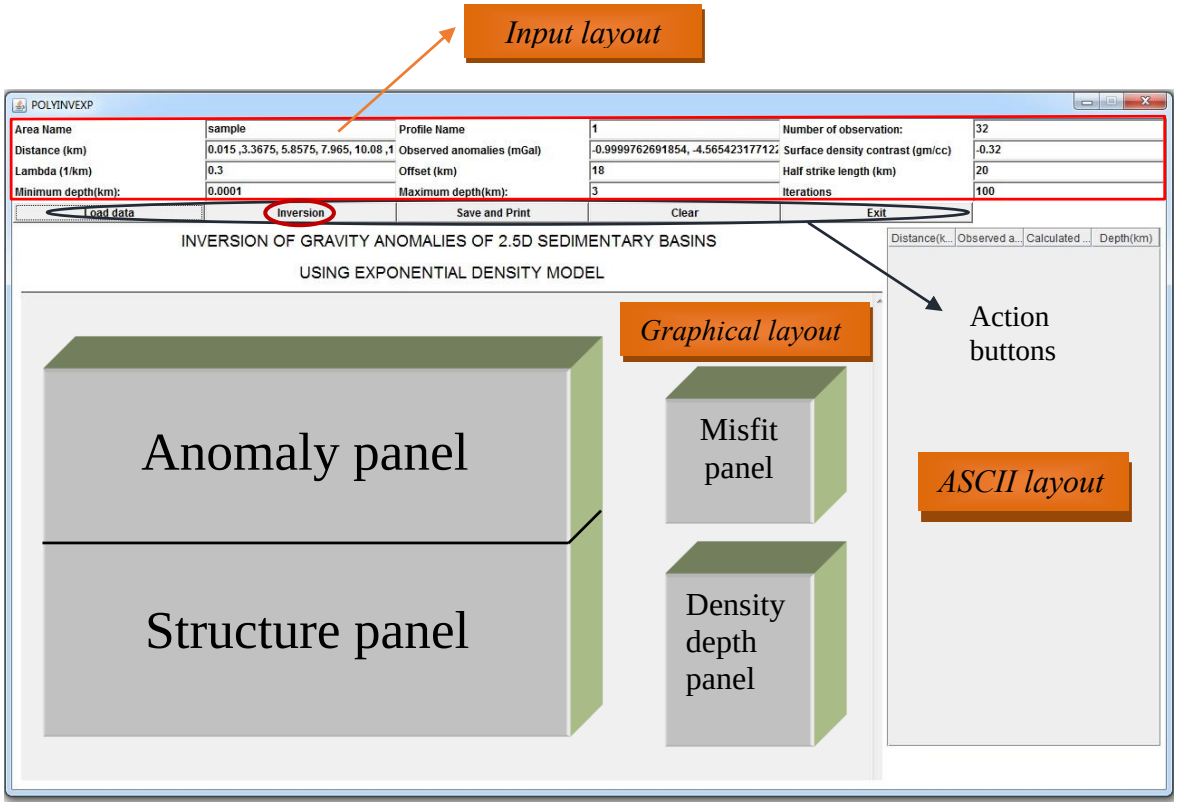


Fig.3.1 View module of POLYINVEXP

Upon invoking the batch file of the code, POLYINVEXP.BAT, the view module consisting of the input, graphical and ASCII layouts appears on the monitor as shown in Fig. 3.1. The user has the choice of entering the input data to the code by either an excel sheet or using respective fields of the input layout. If the choice of entering input data in an excel sheet is

opted the file can be loaded into the software by 'Load data' button of input layout. To avail such an option executable jar file jxl-2.6 has to be downloaded and saved in the executable folder of the code. As in the case of POLYMODEXP (presented in section 2.3.1), this code requires information pertaining to the area name, profile name, number of observations on the profile, distance to observation (km), observed gravity anomaly (mGal), surface density contrast (g/cm^3), Lambda (km^{-1}), profile offset (km), half-strike length (km), minimum and maximum permissible depths of the interface (km), and the number of iterations to perform. Once input data is loaded the user activates the 'Inversion' action button (Fig. 3.1) for data optimization. The business logic of the code initializes the model space by setting the two coefficients of regional background to zero, calculates model response, constructs and solves normal equations for improvements in unknown parameters and updates both model geometry and regional background without the interference of the user. The anomaly panel of the graphical layout displays the observed, modeled gravity anomalies and regional component, structural panel portrays model space, misfit and density-depth panels present variations in data misfit and density-depth in animated forms, respectively (Fig. 3.1). Finally, ASCII layout shows the interpreted results in a tabular form. The 'Save and print' action button facilitates the user to save the output in html format and then to print.

3.4 Applications

The methodology and code of the proposed inversion are applied first on a set of gravity anomalies produced by a synthetic model of strike-limited sedimentary basin in the presence pseudorandom noise. The noisy anomalies are analyzed independently both with and without regional background. In either case, the interpreted results are compared with those of assumed ones. The technique is also applied to invert real-field gravity anomalies measured over a set of sedimentary basins in western Turkey. In both examples EDM explains the density contrast variation of sedimentary rocks with depth.

3.4.1 Synthetic example

The noisy gravity anomalies shown in Fig. 2.7a (solid line in blue) at 18 km offset across the strike of a sedimentary basin are treated here as observed gravity anomalies to run the proposed inversion for recovering basement topography. It is to note that the same anomaly has been analyzed in section 2.3.1 by automatic modeling for estimating basement configuration of the model. The purpose of considering the same anomaly in the present case for inversion is to examine whether both strategies can yield similar interpretations or

not. As in the case of automatic modeling, the constants of EDM are held unaltered during the process of inversion.

The proposed inversion technique was run on the noisy anomalies (solid line in blue in Fig. 2.7a) by setting 18 km as offset parameter, and 0.05 mGal as misfit threshold for program termination. Though a total of 50 iterations are specified as default to run the inversion, for present case the code had taken mere 12 iterations and got terminated because the resulting misfit norm fell below the predefined threshold of 0.05 mGal. The differences between the measured and theoretical gravity anomalies corresponding to the minimum data misfit are shown in Fig. 3.2a. A maximum data residual of -0.07 mGal between the two anomalies was observed on the profile at the 15th km. Fig. 3.2b depicts how the data misfit has changed from the start of the inversion to the 12th iteration. The magnitude of the data misfit was 2.089 mGal for the initial structure, dropped down rapidly to 0.168 mGal after the 4th iteration, and beyond which it decayed gradually (Fig. 3.2b) before the inversion process was terminated at the end of the 12th iteration. It is observed from Fig. 2.8b and 3.2b that the decay of data misfit was more rapid with automatic modeling when compared to inversion.

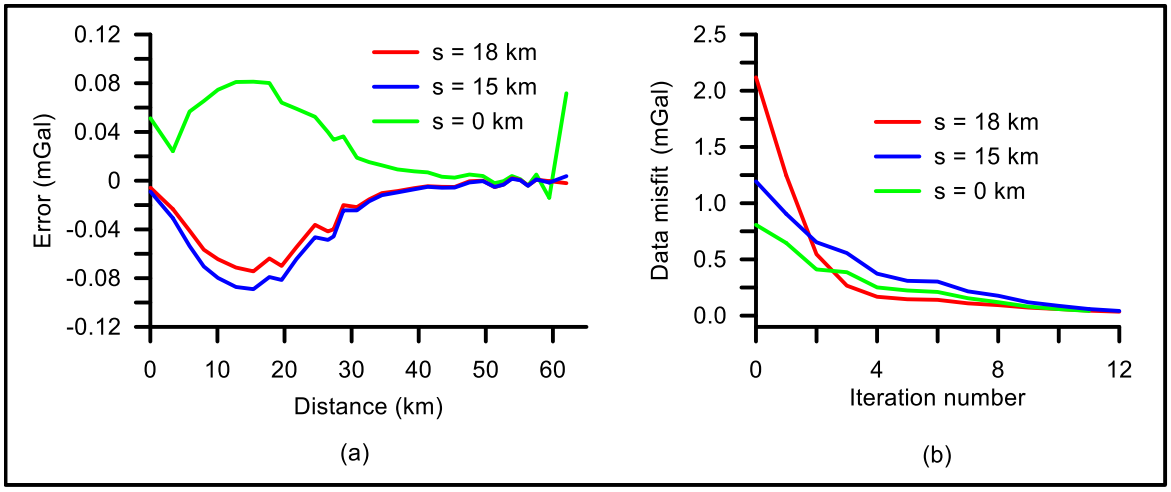


Fig. 3.2 (a) Errors between observed and theoretical gravity anomalies for different profile offsets used in inversion, (b) changes in data misfits against iteration for different profile offsets, synthetic example.

Fig. 3.3a shows the gravity response of the derived structure corresponding to the minimum data misfit. To judge the goodness of fit, the observed noisy anomaly shown in Fig. 2.7a was also plotted in Fig. 3.3a to the same scale. The theoretical gravity anomaly realized from inversion after the 12th iteration coincides excellently well with the observed anomaly (Fig. 3.3a). The estimated structure from inversion of anomalies closely replicates the assumed structure (Fig. 3.3b) despite the fact that the anomaly used in the inversion was noisy.

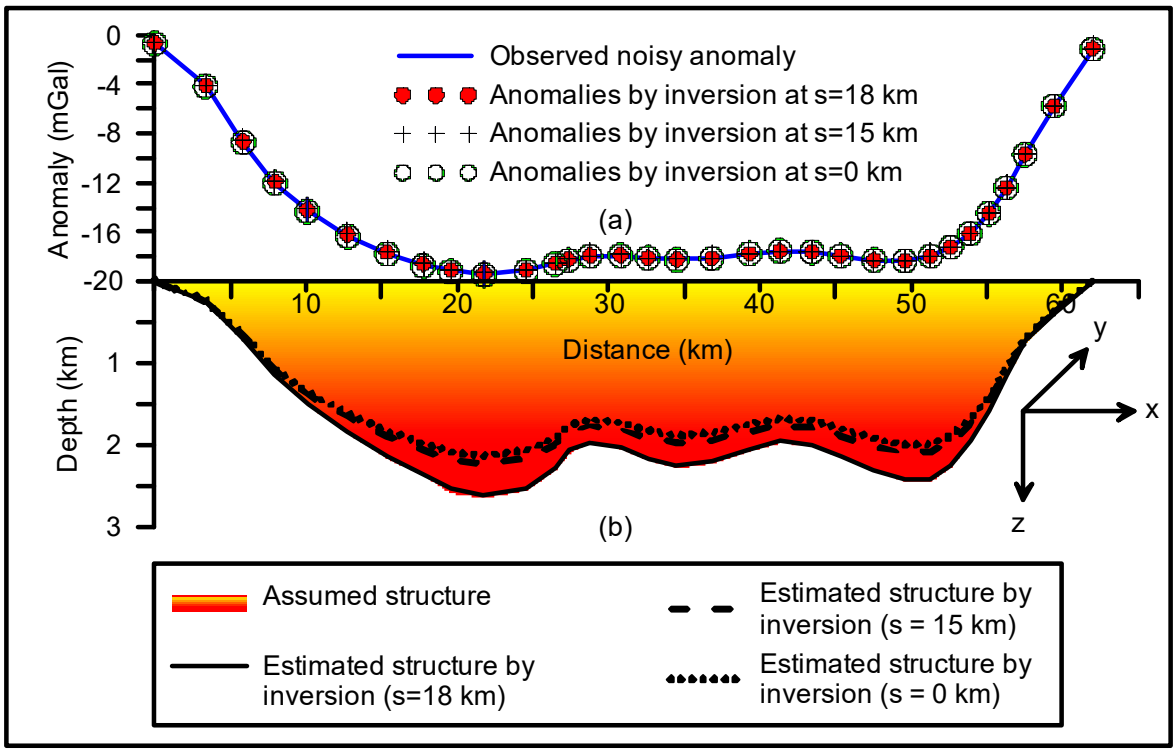


Fig. 3.3 (a) Observed, and theoretical anomalies by optimization at different profile offsets, (b) presumed and inferred structures for different offsets, synthetic example.

Fig. 3.4 shows the errors between the presumed and inferred depths of the structure (expressed in %) at different observer locations over the profile length. The maximum deviation between the two structures is noticed on the profile at 27.33 km, where the structure is underestimated by 2.09%. The errors in estimated depths at all remaining observations on the profile are found just within ±1.72%. Clearly, such marginal errors in estimated depths are acceptable because the anomalies used in the inversion contain random noise.

In the next phase, inversion was performed on the same noisy anomaly (shown in Fig. 3.3a), independently, by considering different profile offsets i.e., 0 km and 15 km as in the case of automatic modeling. This exercise has been carried out to assess the role of profile offset on the inversion. The algorithm had performed 11 iterations when the profile offset is set to zero, and 12 iterations in case of 15 km offset. In either case, the magnitude of data misfit had shown progressive decay with the increase in iteration number (Fig. 3.2b). The start value of data misfit corresponding to the initial model was 0.8075 mGal in case of zero offset, and 1.193 mGal in case of 15 km offset. The magnitudes of data misfits in both cases fell below the prescribed threshold of 0.05 mGal at the end of respective concluding iterations. The errors between the measured and theoretical gravity anomalies after the

inversion were found within ±0.08 mGal (Fig. 3.2a). Overall, the misfit corresponding to 18 km profile offset was decayed much more rapidly with iteration when compared to the other two cases (Fig. 3.2b).

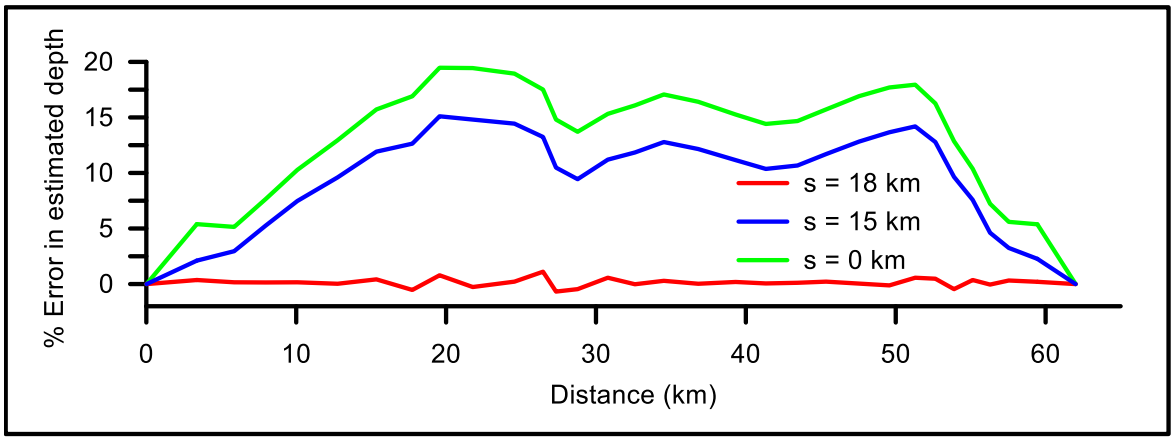


Fig. 3.4 Errors (%) between assumed and estimated depths for different profile offsets used in inversion, synthetic example.

In spite of the fact that the theoretical gravity anomalies realized with zero and 15 km profile offsets explain the observed anomalies more or less in the same way (Fig. 3.3a), the estimated structures do not coincide with each other (Fig. 3.3b). The error plot (%) shown in Fig. 3.4 reveals that the inferred structure in either case is undermined by as much as 20% in case of zero profile offset and 15-16% in case of 15 km offset in depocentral regions of the basin.

To test the efficacy, the present inversion technique was applied to analyze the anomalies in the presence of a variable regional component that was presumed to be retained in the structure anomaly. A linear term described with a set of randomly chosen coefficients (Table 3.1) was added to the anomalous field shown in Fig. 3.3a before inversion was performed. Fig. 3.5 shows how the assumed regional field varies over the length of profile across the strike of the structure.

Table 3.1
Assumed and computed coefficients of regional component, case of Synthetic example

Offset	**Assumed**		**Estimated**	
(km)	f_0(mGal)	f_1(mGal/km)	f_0(mGal)	f_1(mGal/km)
18	-0.33	-0.023	-0.429	-0.016
15			-0.365	-0.015
0			-0.302	-0.0143

The combined anomaly (regional + residual + pseudorandom) is shown in Fig. 3.6a as a solid line in black. The inversion technique was applied on this combined noisy anomaly, again, choosing different profile offsets viz., 18 km, 15 km and zero km.

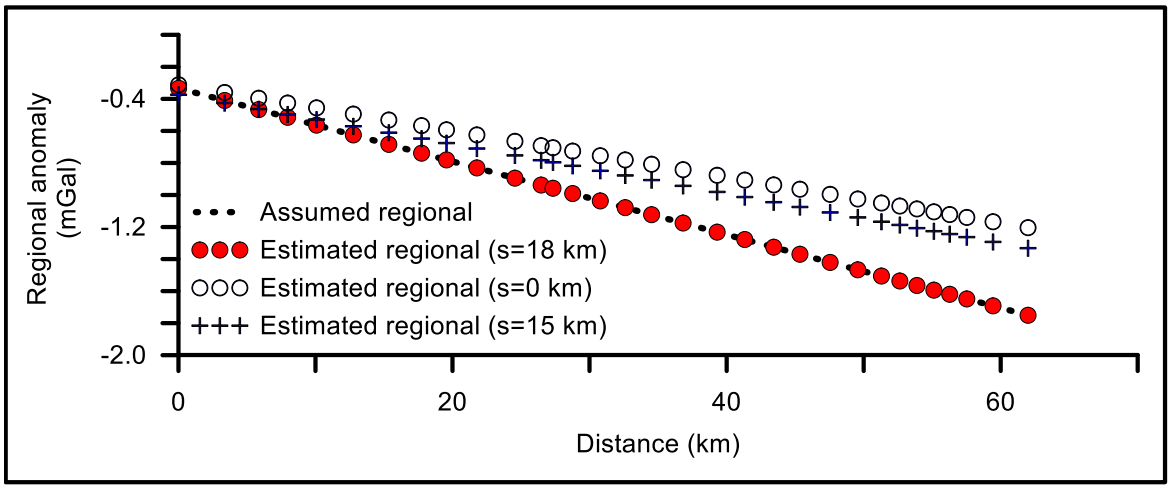


Fig. 3.5 Assumed and estimated regional components for different profile offsets used in inversion, synthetic example.

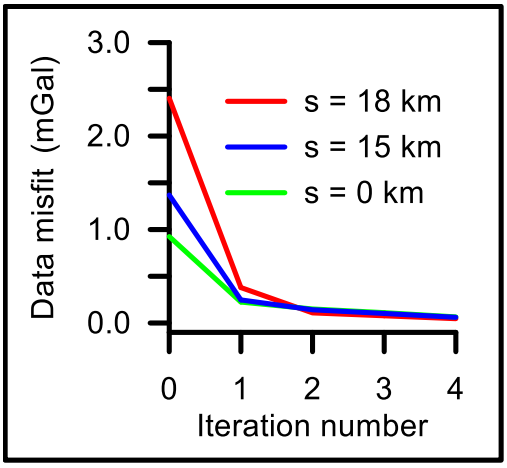


Fig. 3.6 Misfit variation for different profile offsets used in the inversion of combined gravity anomaly, synthetic example.

The data misfit in each case after the 4th iteration had attained a magnitude smaller than the specified threshold, thereby the algorithm got terminated. Though the initial data misfits for different profile offsets are different (2.405 mGal for 18 km offset, 1.3685 mGal for 15 km offset, and 0.9246 mGal for zero km offset), the rate at which the misfit decays with iteration remains more or less the same in all the cases beyond the 2nd iteration (Fig. 3.6). The coefficients of regional component estimated by the algorithm for different profile offsets are given in Table 3.1.

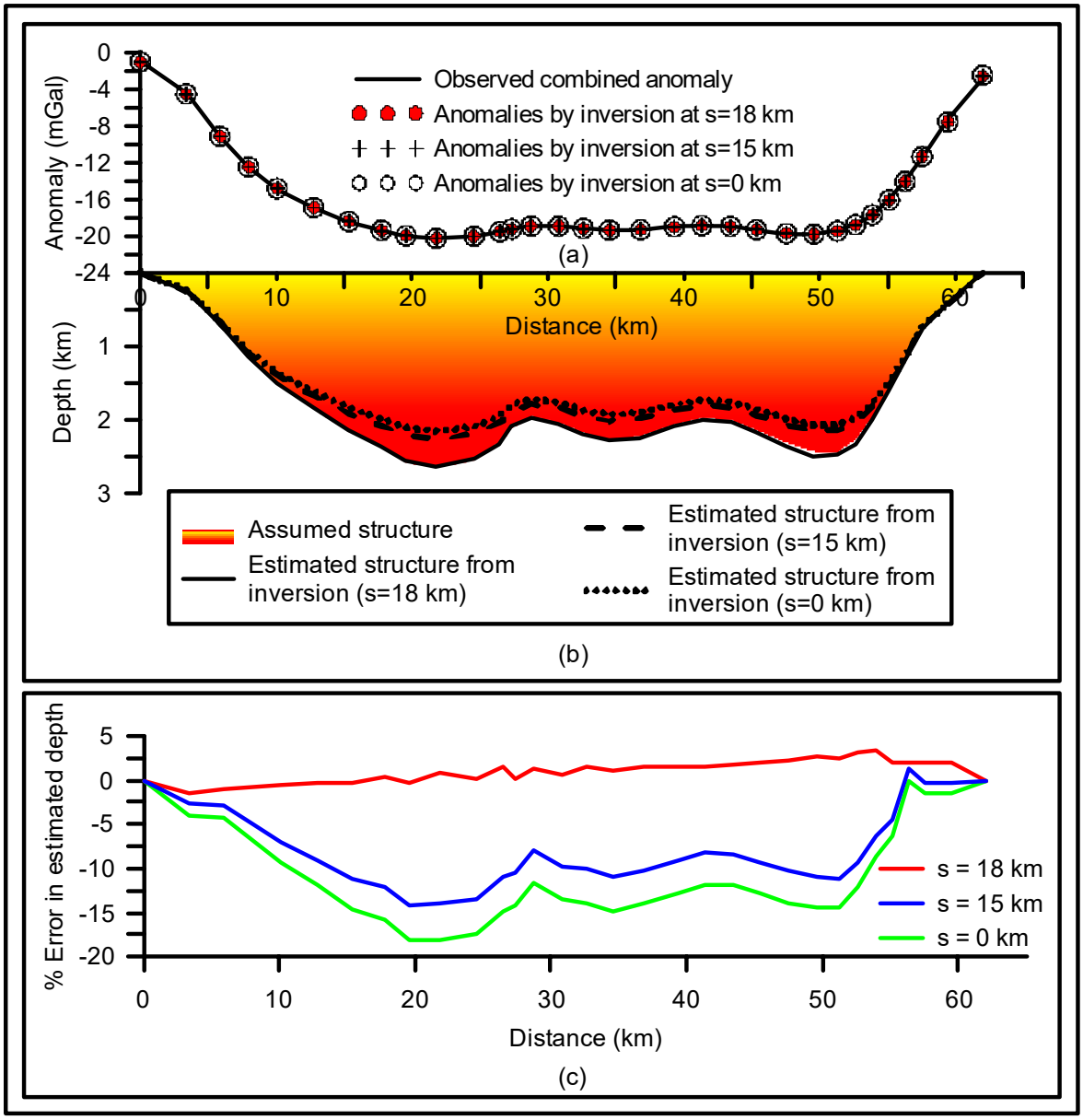


Fig. 3.7 (a) Observed and theoretical combined gravity anomalies by inversion at different profile offsets, (b) assumed and estimated structures for different profile offsets, (c) error analysis between the assumed and estimated structures for different offsets, synthetic example (Chakravarthi et al., 2017a).

Fig. 3.5 clearly shows that the estimated regional with the optimum profile offset of 18 km closely fit the assumed regional, whereas the same is not repeated with either 15 km offset or zero km offset. By and large, the recalculated anomalies of the model space after the 4th iteration for different profile offsets equally fit the observed anomaly (Fig. 3.7a), however, the estimated structures do not comply well with each other (Fig. 3.7b). Setting up of the profile offset to 18 km in the inversion had yielded a structure that is slightly overestimated by 3.27% (maximum error) within the distance observation range [30 km, 55 km] (Fig. 3.7c). On the other hand, the choice of 15 km and zero km offsets in the anomaly inversion had

resulted structures that are underestimated throughout profile almost at all observations (Fig. 3.7b). A maximum error of 14.28% was observed in the estimated depth at the 20th km on the profile when the offset was set to 15 km, and 18.19% in case of zero profile offset at the same observation (Fig. 3.7c).

To sum up, the proposed inversion successfully recovers the assumed structure even when the structure anomaly contain significant level of pseudorandom noise, however, proper offset needs to be considered in the inversion. Furthermore, it is proved that the proposed technique is insensitive to linear regional background in the sense that the presence of any unremoved regional component in the structure anomaly does not affect the interpretation appreciably.

3.4.2 Field example

For real field applications, the proposed inversion technique was applied to decipher basement configurations of the Gediz graben along two selective anomaly profiles (Fig. 2.14 a, c), and along one profile (Fig. 2.14e) in the Büyük Menderes graben. In Chapter-2, the same anomaly profiles are analyzed by the principles of automatic modeling. As in the case of automatic modeling, the derived constants of EDMs of respective grabens are held unaltered during the inversion of respective gravity anomaly profiles. The allowable threshold set for data misfit for algorithm termination in each case was 0.001 times the number of observations.

The technique had performed 8 and 4 iterations respectively for profiles 2 and 3 of the Gediz graben. In case of profile XY of the Büyük Menderes graben it took 79 iterations for the convergence. The data misfits for both profiles pertaining to the Gediz graben after the respective concluding iterations fell below the predefined threshold, whereas for the profile, XY, the damping factor after the 79th iteration had attained a large value. The data misfits correspond to the starting and optimum estimated models for all the three profiles under consideration are given in Table 3.2. Overall behavior of the data misfit with iteration for each case is shown graphically in Fig. 3.8. The theoretical gravity anomalies obtained through inversion at the end of respective closing iterations are shown in Figs. 3.9a, c and e respectively for the profiles 2, 3 and XY.

In all cases, it is noticed that the theoretical gravity anomalies after the concluding iterations closely mirrored the measured anomalies. The optimum structure models corresponding to

minimum data misfits (Table 3.2) are shown in Fig. 3.9b, d and f along with the models reported by Sari and Şalk (2002) from 2D modeling. The estimated coefficients of regional components in each case are given in Table 3.3 and shown in Figs. 3.9a, c and e. The interpreted basement structures of the Gediz and Büyük Menderes grabens along the three profiles from present inversion (Figs. 3.9 b, d and f) comply reasonably well with those derived from automatic modeling (Figs. 2.14 b, d and f).

Table 3.2
Data misfits for initial and optimum models by inversion
Gediz and Büyük Menderes grabens, western Turkey (Chakravarthi et al., 2017)

Graben/Profile No.	**Data misfit (mGal)**	
	Starting	Ending
Gediz/2	1.585	0.069
Gediz/3	0.674	0.038
Büyük Menderes/XY	1.539	0.144

Table 3.3
Coefficients of linear regional obtained from inversion
Gediz and Büyük Menderes grabens, western Turkey (Chakravarthi et al., 2017)

Graben/Profile No.	**Coefficients**	
	f_0 (mGal)	f_1(mGal/km)
Gediz/2	-4.021	0.1102
Gediz/3	0.062	-0.0021
Büyük Menderes/XY	0.231	0.0041

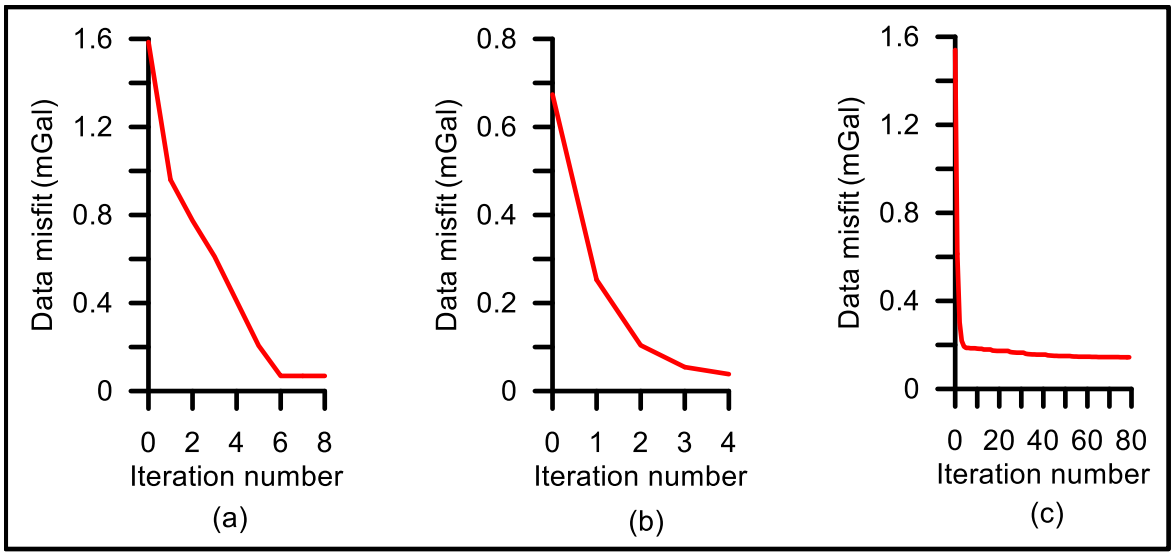


Fig. 3.8 Behaviour of data misfits with iteration number (a) profile 2, (b) profile 3 (Gediz graben), (c) profile XY (Büyük Menderes graben), western Turkey.

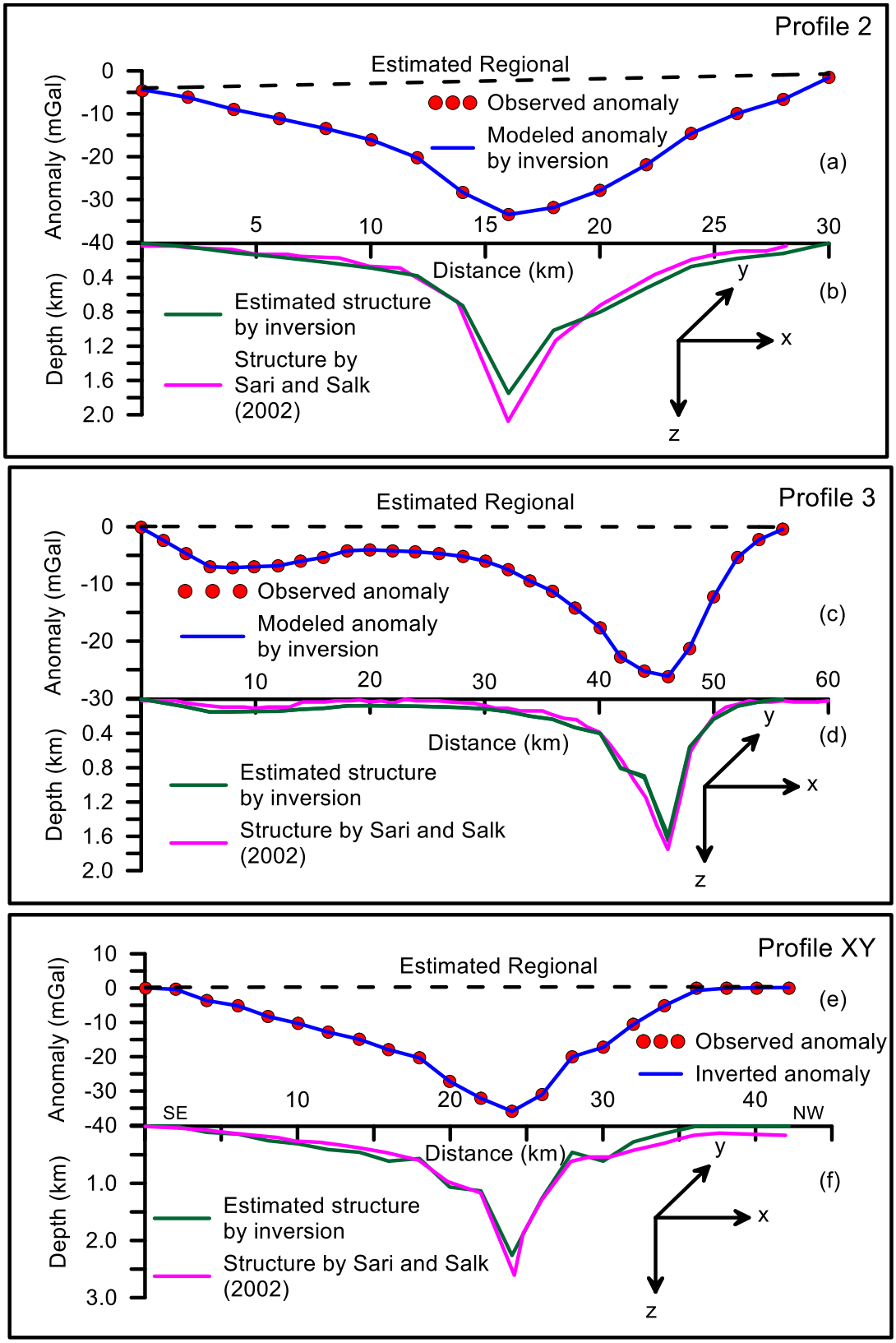


Fig. 3.9 Observed, modeled gravity anomalies (a, c, and e), and estimated structures (b, d, and f) from inversion - Profile 2 and 3, Gediz graben and profile XY, Büyük Menderes graben, western Turkey. Interpreted 2D models by Sari and Şalk (2002) are also shown for comparison (Chakravarthi et al., 2017a).

Table 3.4 compares the maximum depths of the basement inferred from the present inversion and by automatic modeling technique along the three selected profiles against the reported depths by Sari and Şalk (2002).

Table 3.4
Estimated basement depths (at depocentres) from present inversion and automatic modeling techniques, Gediz and Büyük Menderes grabens, western Turkey (after Chakravarthi et al., 2017)

Graben/Profile No.	Estimated depth from modeling (km)	Estimated depth from inversion (km)	Reported depth by Sari and Şalk (2002) (km)
Gediz/2	1.91	1.75	2.07
Gediz/3	1.65	1.60	1.75
Büyük Menderes/XY	2.26	2.32	2.60

All the interpreted structural models estimated from proposed inversion technique reinstates the fact in line with the automatic modeling that the anomaly interpretation of a finite strike basin by a 2D technique would yield a structure that has shallower basement on the shoulders and relatively deeper basement at the depocentres when compared to actual ones.

3.5 Results and discussion

i) Using the principles of optimization, a new space domain technique is developed to interpret sedimentary basin gravity anomalies taking into account 2.75/2.5 dimensionality for the structure with exponential density variation. Based on the methodology, a software, POLYINVEXP, is developed and presented.

ii) The proposed inversion treats the geometry of a sedimentary basin as a finite-strike vertical polygon with exponential density variation to realize forward modeling.

iii) The present technique simultaneously solves the depth parameters of a sedimentary basin and two constants of linear regional field from the observed anomalies.

iv) Applicability of the proposed optimization and software is illustrated with theoretical and field gravity anomalies.

v) In case of the theoretical example, the noisy anomalies attributable to a strike-limited sedimentary basin are interpreted both with and without regional background. In both cases, the technique has yielded structures which are almost identical to the assumed one. It is also demonstrated with the synthetic example that choice of

inappropriate profile offset in the inversion of gravity anomalies would yield unviable structure models.

vi) The inferred basement configurations of the Gediz and Büyük Menderes grabens in western Turkey from the inversion of observed gravity anomalies are found consistent with the reported information, hence, the practical applicability of the technique is established.

CHAPTER
FOUR

3D Automatic modeling of gravity anomalies - model simulation by multiple polygonal cross-sections coupled with EDM *

4.1 General

Although the gravity method is one among the age old geophysical methods, it still enjoys its pivotal role to address a variety of geologic problems associated with the exploration of natural resources, various geodynamic/tectonic processes, etc. In regional and hydrocarbon explorations, the gravity method provides vital information on the disposition and orientation of sedimentary basins, even when the basins are concealed under thick pile of basalt cover (Chakravarthi et al. 2007c). The information gained from a gravity survey over a region is very crucial and informative to plan and execute seismic surveys in a more systematic manner.

Forward modelling i.e., computation of gravity response of a prescribed model space described with known set of shape and size parameters forms an important exercise in automatic modeling and inversion studies. Talwani and Ewing (1960) proposed an ingenious method to compute the gravity response of a 3-D source at an external point, wherein the gravity effects of several horizontal polygonal laminas representing the outline of the source volume were computed using a constant density and then numerically integrated to obtain the total gravity effect of the source. However, it becomes difficult to use such a method in automatic modelling schemes to model the gravity anomalies resulted from sedimentary basins. This is because the modelling result would be severely affected when the depth ordinates of vertices of several laminas are kept unchanged while allowing only the horizontal coordinates of vertices to change during the

---

* *Published in Pure and Applied Geophysics, 2019, 176, 2497-2511.*

process of analysis (Rao et al., 1999; Mallesh et al., 2019). Nagy (1966), Banerjee and Das Gupta (1977), and Tsoulis (2000) have presented equations to compute the gravity anomalies due to parallelepipeds using uniform density, and for polyhedral bodies, Paul (1974), Okabe (1979), Pohanka (1988), Holstein (2002), and D'Urso (2014b) have provided formulae.

3D techniques to analyze the gravity anomalies of sedimentary basins treating the sediment density as uniform are available (see for e.g., Cordell and Henderson 1968; Pilkington and Crossley 1986; Murthy et al. 1990; Comacho et al. 1997; Debeglia and Corpell 1997; Zhdanov and Cai (2013). However, the practical utility of the enlisted 3D modeling schemes fall short to analyze the gravity anomalies if geologic settings warrant the use of non-uniform density. On the other hand, Hansen (1999), Holstein (2003), D'Urso (2014a) have presented formulae to calculate the gravity anomalies due to a polyhedral source using linear density model, Chakravarthi et al. (2002), García-Abdeslem (2005) have derived analytical gravity expressions due to a right rectangular prism using parabolic and cubic polynomial density models, and Wu and Chen (2016) have presented a formula using general polynomial models. Gokula and Sastry (2015) have adopted the parabolic density model to obtain a formula to realize forward modeling of a vertical pyramid that is bounded on the top and bottom by planar surfaces. Nasuti and Ardestani (2007) have proposed a forward modeling scheme to calculate the gravity anomalies of 3D sedimentary basins using a quadratic density model.

Rao et al. (1990b), Gallardo-Delgado et al. (2003), Feng et al. (2015) have presented 3D algorithms to estimate the depths of basement interfaces from the observed gravity anomalies using quadratic density-depth models. Chakravarthi (2003), Chakravarthi and Sundararajan (2004a) have devised automatic schemes using parabolic density models to interpret the gravity anomalies resulting from sedimentary basins. Later, Isik and Senel (2009), and Bal and Kara (2012) have adopted the 3D interpretation methodologies proposed by Chakravarthi (2003) and Chakravarthi and Sundararajan (2004a) to analyze the gravity anomalies of the Büyük Menderes basin and the Salt Lake region of Turkey, respectively.

Many of present day 3D interpretation techniques that make use of exponential density models to analyze the gravity anomalies are being developed in the wave number domain because of the simple fact that analytical solutions for forward modeling are unavailable and underivable

in the spatial domain. For this reason Granser (1987), Chai and Hinze (1988), Feng et al. (2015), Pham et al. (2018) have performed the exercise of forward modeling of gravity anomalies of a 3D source in the frequency domain and transformed the anomalies back to spatial domain for further analysis. It is obvious that such a procedure of transforming the anomalies from the frequency to spatial domain invariably experiences truncation errors as the data window is always limited (Mallesh et al., 2019).

In this chapter, a technique in the spatial domain to calculate gravity anomalies of 3-D sources among which the density contrast obeys exponential decrease with depth is presented. This technique judiciously combines both analytic and numeric approaches. Followed by this a 3D method using the principles of automatic modelling (Chakravarthi et al., 2013a) and related software are developed to recover basement structures of sedimentary basins from observed gravity anomalies within the predefined convergence criteria. The applicability of modeling technique is shown by analyzing the gravity anomalies produced by a synthetic structure in the presence of pseudorandom noise and then over a real world gravity data pertaining to the Los Angeles basin, California. In either case, the density contrast varies exponentially with depth.

4.2 Forward modeling – Theoretical considerations

In Cartesian co-ordinate system, let the z-axis be positive vertically downwards with the x-axis running transverse to it (Fig. 4.1). The y-axis is perpendicular to the xz plane and directed inwards along which the strike of the model space is scaled (Fig. 4.1). The gravity anomaly, $\Delta g\ (0,0,0)$, of such a model at a point, $P(0, 0, 0)$, can be expressed as (Mallesh et al., 2019)

$$\Delta g\ (0,0,0) = G \int_{v} \frac{\Delta\rho(z)\mathrm{z}\ \mathrm{d}v}{(x^2 + y^2 + z^2)^{3/2}}, \tag{4.1}$$

where (x, y, z) are the source co-ordinates of an elementary volume, dv, within the structure. The density contrast $\Delta\rho(z)$ of sediments within the source volume varies according to equation (1.12).

Upon describing the model space by a stack of multiple vertical laminas, each one with unit thickness (Murthy and Rao, 1985), equation (4.1) can be expressed as

$$\Delta g\ (0,0,0) = G \int_{Y1}^{Y2} \left[\int_{s} \frac{\Delta\rho(z)\mathrm{z}\ \mathrm{ds}}{(x^2+y^2+z^2)^{3/2}} \right] dy, \tag{4.2}$$

where, $Y1$ and $Y2$ are the limits of the model space as measured from the point of calculation and the integration is carried out numerically using the trapezoidal rule. Substituting equation (1.12) and applying Stokes' theorem, equation (4.2) becomes

$$\Delta g\ (0,0,0) = G\Delta\rho_0 \int_{Y1}^{Y2} \left[\oint_{z} \left[\frac{xze^{-\lambda z}}{(y^2+z^2)(x^2+y^2+z^2)^{1/2}} \right] dz \right] dy, \tag{4.3}$$

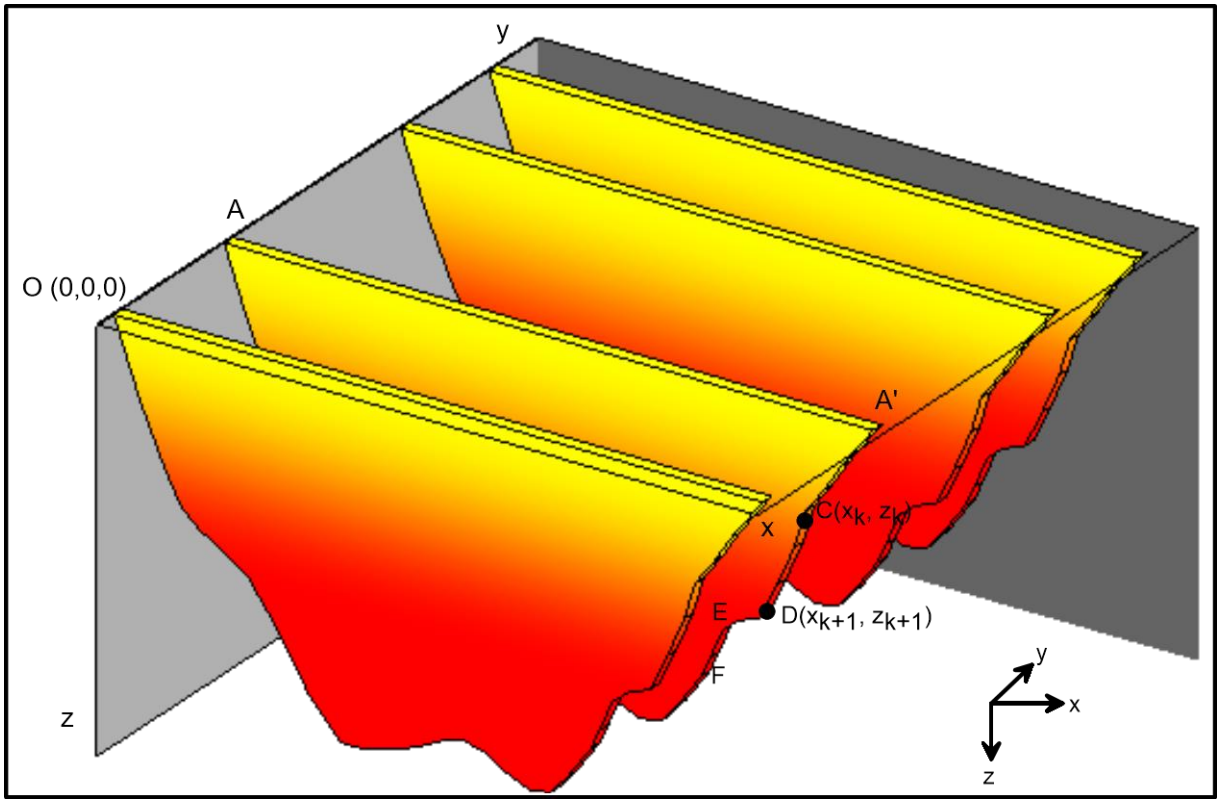


Fig. 4.1 Schematic representation of a 3D sedimentary basin by a stack of vertical polygonal cross-sections in Cartesian coordinate system.

To solve the inner integral in equation (4.3), x needs be expressed in terms of z. This could be realized by approximating the outline of a vertical lamina by a multifaceted polygon AA'CDEF…(as shown in Fig. 4.1). If θ is the angle made by the side CD with the horizontal passing through the vertex C (parallel to x-axis) then

$$x = x_k - z_k cot\theta + zcot\theta, \tag{4.4}$$

where, (x_k, z_k) are the co-ordinates of the vertex C. Upon substitution of equation (4.4), equation (4.3) takes the form

$$\Delta g\ (0,0,0) = \int_{Y1}^{Y2} \Delta g(0, y, 0)\ dy, \tag{4.5}$$

where,

$$\Delta g(0, y, 0) = G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} \frac{(\ x_k - z_k cot\theta + zcot\theta)\ ze^{-\lambda z}}{(y^2 + z^2)\ [(x_k - z_k cot\theta + zcot\theta)^2 + y^2 + z^2]^{\frac{1}{2}}} \mathrm{d}z. \tag{4.6}$$

Here, N is the number of faces bounded by a polygon in the xz-plane. Upon simplification, equation (4.7) can be rewritten as,

$$\Delta g(0, y, 0) = G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} \frac{(\ x_k sin\theta + \overline{z - z_k}\ cos\theta)\ ze^{-\lambda z}}{(y^2 + z^2)\ (A1 + A2 + A3)^{\frac{1}{2}}} \mathrm{d}z, \tag{4.7}$$

Here, z_{k+1} is the depth ordinate of the vertex D. Also,

$$A1 = \overline{x_k^2 + y^2 + z^2}\ sin^2\theta$$

$$A2 = \overline{z - z_k}^2\ cos^2\theta$$

$$A3 = x_k \overline{z - z_k}\ sin2\theta$$

$$sin\theta = \frac{z_{k+1} - z_k}{[(z_{k+1} - z_k)^2 + (x_{k+1} - x_k)^2]^{1/2}}$$

$$cos\theta = \frac{x_{k+1} - x_k}{[(z_{k+1} - z_k)^2 + (x_{k+1} - x_k)^2]^{1/2}}$$

Here, x_{k+1} is the x coordinate of the vertex D. The coordinates of the vertices are expressed in km and the density contrast in g/cm^3, which results the gravity anomalies in mGal. Equation (4.7) is to be evaluated numerically because no closed form analytic solution could be derivable in the spatial domain. It is to note that the vertices of a polygon, whose coordinates are scaled with reference to the point of calculation, are covered sequentially in the clockwise direction (Chakravarthi et al., 2017) with respect to the increasing y- axis. In case the vertices are covered

anticlockwise, the magnitude of anomalous field remains the same but its sign changes. For a horizontal segment of a polygon, the integral in equation (4.7) takes the form

$$\int_{z_k}^{z_{k+1}} \frac{ze^{-\lambda z}}{(y^2+z^2)}\mathrm{d}z, \tag{4.8}$$

which ultimately becomes zero. Furthermore, equation (4.7) has a singularity point for an outcropping cross-section at $y = 0$. In such a case, the corresponding cross-section is treated as a 2.5D polygon with unit strike length ($\Delta Y = 2Y = 1$) within which the density contrast obeys variation according to equation (1.12). The gravity anomaly of such a source can be expressed as (Mallesh et al., 2019)

$$\Delta g_k = 2G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} e^{-\lambda z}\, tan^{-1} \frac{cos\theta}{2\left(0.25\, sin^2\theta + \left(\frac{z}{\Delta Y}\right)^2\right)^{1/2}}\mathrm{d}z \tag{4.9}$$

The gravity contributions of several representative polygons are calculated and finally integrated over the strike length of the model space to obtain its total gravity anomaly at the point of calculation.

4.2.1 Accuracy assessment of forward modeling

The validity of the proposed forward modeling is justified by computing and comparing the model gravity response of a synthetic homogeneous sedimentary basin (geometry is shown in Fig. 4.2a) obtained from equation (4.5) with the anomalous field realized through the method of Talwani and Ewing (1960). In this case, the maximum thickness of sedimentary basin is 1.42 km with a presumed density contrast of -0.32 g/cm^3.

The gravity anomalies of the structure was calculated at 825 observations using equation (4.5) and shown in Fig 4. 2b as solid lines in red. The anomalous field obtained from the method of Talwani and Ewing (1960) at the same locations are shown in Fig. 4.2b by solid lines in blue. The closeness of fit between the anomalies realized from the proposed method and the ones from the method of Talwani and Ewing (1960) proves the accuracy of the proposed forward modeling.

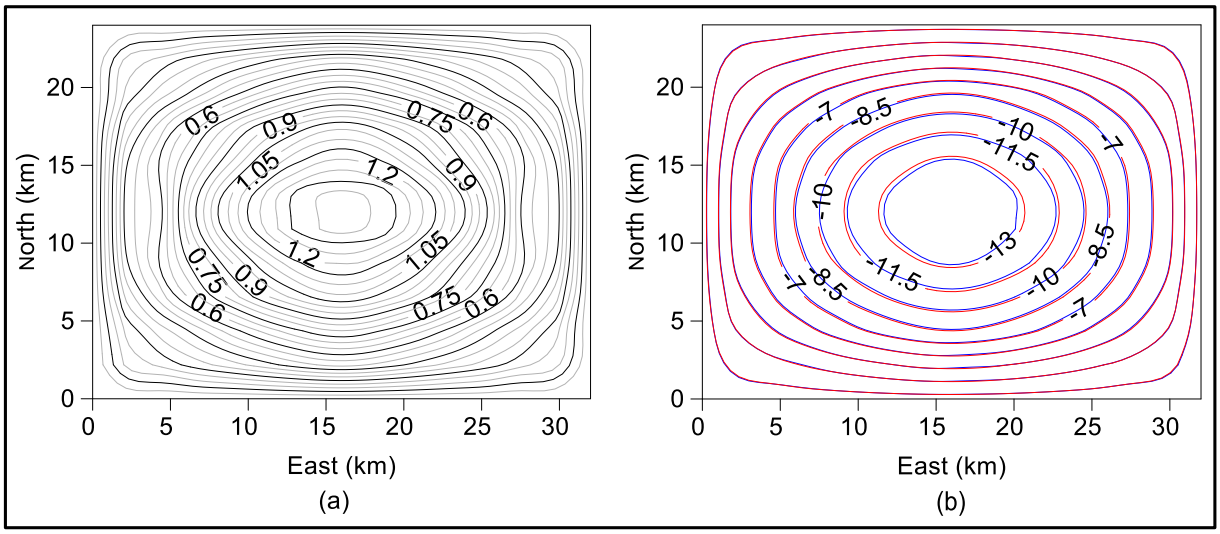


Fig. 4.2 (a) Synthetic homogenous sedimentary basin, (b) model gravity response obtained from the present method and the one by Talwani and Ewing (1960).

4.3 Modeling of gravity anomalies

Let N_x and N_y be the number of observations along the x-axis and y-axis, respectively. It is presumed that observed gravity anomalies over the topography are available over and beyond the basin boundary so that the relief of the basin all along its periphery becomes zero. The problem of gravity interpretation then becomes to estimate $(N_x - 2)*(N_y - 2)$ depth parameters from $N_x* N_y$ observed gravity anomalies. Furthermore, known information on the basement depths, if any, available at isolated nodes/observations (for e.g., from boreholes etc.) can be specified as constraints in the modeling. To start with, the gravity anomaly at each observation is presumed as being produced by a slab of infinite horizontal extent, in which the density contrast also obeys exponential decrease following equation (1.12). Such approximation ensures one to obtain the initial parameters of the model in close proximity to the true ones. The thickness of the slab (z_B) and the gravity anomaly produced by it (g_B) are related to each other by equation (2.9). This equation forms the basis to estimate the initial depths, $z_{(x_i,y_j)}$, of a density interface at any observation, (x_i, y_j). Substituting the observed gravity anomaly, $\Delta g_{(x_i,y_j)}$, in place of g_B in eq. (2.10),

$$z(x_i, y_j) = z_B = \frac{-1}{\lambda} log\left(1 - \frac{\lambda \Delta g_{(x_i,y_j)}}{2\pi G \Delta \rho_0}\right). \tag{4.10}$$

Subsequently, equation (4.5) calculates the model gravity response of a sedimentary basin, $\Delta g_{M(x_i,y_j)}$, at plurality of observations/nodes, (x_i, y_j), for i = 1, 2,…, N_x and j = 1, 2,…, N_y. The difference between the observed ($\Delta g_{(x_i,y_j)}$), and model gravity ($\Delta g_{M(x_i,y_j)}$) anomalies is quantified by data misfit defined by (Mallesh et al., 2019)

$$J = \sqrt{\frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left[\Delta g_{E(x_i,y_j)}\right]^2}{N_x * N_y}}, \tag{4.11}$$

where,

$$\Delta g_{E(x_i,y_j)} = \Delta g_{(x_i,y_j)} - \Delta g_{M(x_i,y_j)}. \tag{4.12}$$

The depth estimates of the basin are improved, automatically, in an iterative approach based on the equation (Cordell and Henderson 1968; Blakely 1995; Mallesh et al., 2019)

$$z^{k+1}_{(x_i,y_j)} = \frac{\left[\Delta g_{(x_i,y_j)} * z^{k}_{(x_i,y_j)}\right]}{\Delta g^{k}_{M(x_i,y_j)}}. \tag{4.13}$$

Here, k stands for iteration number, and $z^{k+1}_{(x_i,y_j)}$ is the improved depth estimate of the basin at any observation, (x_i, y_j). The gravity response of updated structure is calculated again using equation (4.5) at all the observations and a new data misfit is obtained. This process goes on in a repetitive manner until one of the termination criteria is fulfilled as explained under section 2.3 of Chapter-2.

4.3.1 POLYMOD3D

Based the modeling methodology described in text, a software named POLYMOD3D, coded in Core Java was developed and presented in Annexure 4-A to analyze the gravity anomalies resulted from three-dimensional density interfaces using a prescribed EDM. The software works on Model-View-Controller architecture (Fig. 2.5) as described in section 2.3.1. The view module of POLYMOD3D appears on the monitor as shown in Fig. 4.3 upon invoking the batch file. Input data can be specified to the software by entering the data in a formatted excel sheet

followed by reading the file to the code by 'Load data' action button. To avail this facility, jar file jxl-2.6, has to be downloaded and saved in the executable folder of the code. Alternatively, user may enter the data directly in respective fields of the input layout (Fig. 4.3).

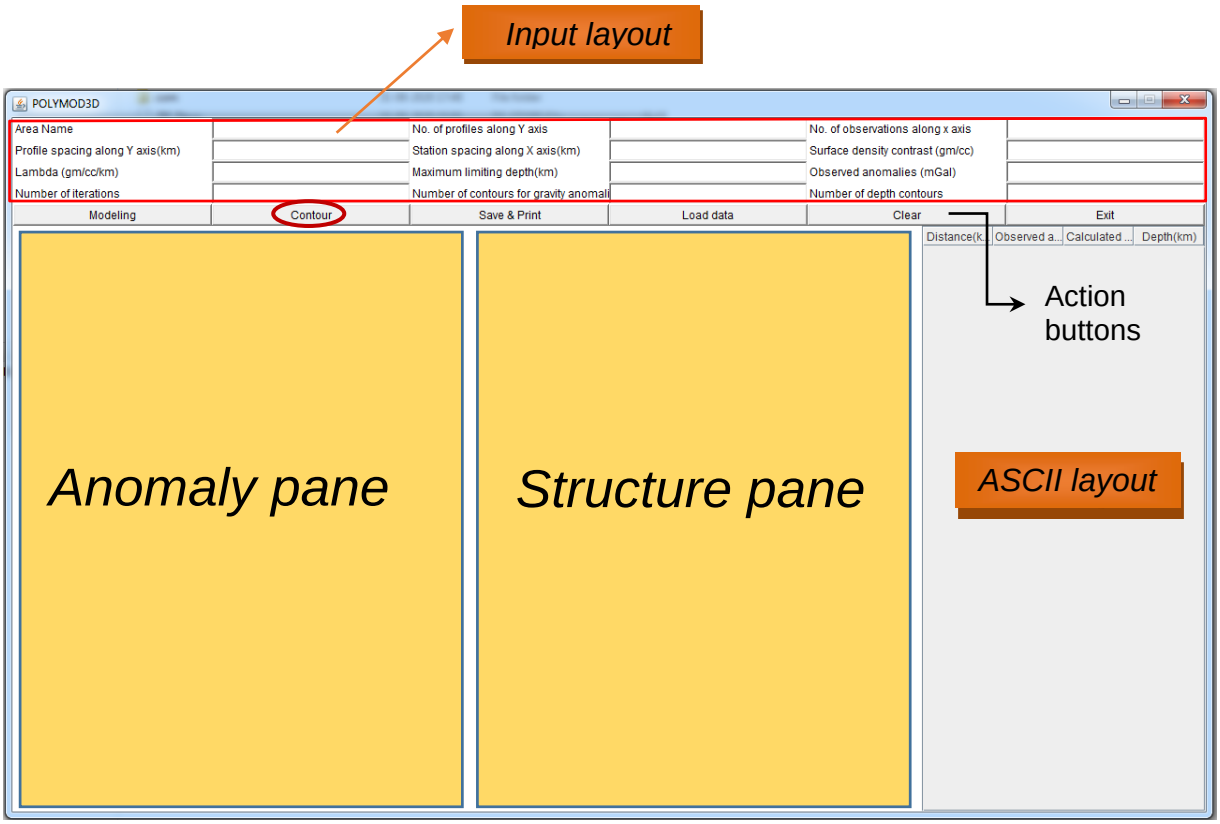


Fig.4.3 View module of POLYMOD3D

The input data consists of area name, number of profiles along the y-axis, number of observations along each profile, profile spacing along the along the y-axis (km), station spacing along the x-axis (km), surface density contrast (g/cm^3), decay constant of EDM (g/cm^3/km), maximum limiting depth (km), observed gravity anomalies covered sequentially from the first profile onwards (mGal), number of iterations to be performed, number of contours required for plotting observed and modelled gravity anomalies, and number of contours required for plotting the basement depth. The user invokes the action button 'Modeling' (located below the input layout) for carrying out automatic modeling within the specified convergence conditions. The graphical layout displays the observed and modeled gravity anomalies, and estimated basement configuration in the form of contour maps. The algorithm of contour plotting (Snyder, 1978) has been used in the software to present contour maps of two-dimensional data. The noteworthy

feature of POLYMOD3D is that it facilitates the user to visualize the changes in model growth (both in terms of shape and depth) and corresponding modelled gravity response in animated forms. Upon the termination of algorithm, the user clicks the 'Çontour' action button to see the final interpreted result in respective contour maps. The ASCII layout shows interpreted results in a tabular form, profile-wise, at the end of the concluding iteration. Using the 'Save & Print' action button the user saves the output file, containing both ASCII data and contour maps, to the disk followed by printing.

4.4 Applications

The method of interpretation described in the text was applied to analyze the gravity anomalies of both synthetic and real-world examples to demonstrate its applicability. In either case EDM explains the variation of density contrast with depth.

4.4.1 Synthetic example

Fig. 4.4a shows in plan view the geometry of a typical intracratonic sedimentary basin with some kind of east-west sagging at the centre. The basement shows progressively increasing dips towards the basin's depocentre with distinct sharp gradients beyond 0.6 km and 1.6 km depths respectively (Fig. 4.4a). The maximum depth to the floor of the basin (2.93 km) is confined between $x \in$ (14.35 km, 17.92 km) and $y \in$ (10.62 km, 13.37 km). Further, the density contrast of the sediments within the basin is varying exponentially with depth defined by $\Delta\rho_0$= -0.32 g/cm^3 and λ = 0.3 g/cm^3/km (equation 1.12). The structure anomaly realized through equation (4.5) is shown in Fig. 4.4b. One can clearly notice from Figs. 4.4a and 4.4b that the faulted basement at large depths, characterized by steep gradients, hardly produce any indicative signatures on the anomaly. Treating the structure anomaly as the observed one, the proposed algorithm is applied over the anomalies to estimate basement depths for which it took 16 iterations before it got terminated. The modeled anomaly at the end of the 16th iteration, being the concluding one, exactly mimics the observed anomaly and the corresponding estimated structure precisely coincides with the assumed structure (this case is not shown for brevity). In the second phase, the structure anomaly is analyzed by the proposed modeling in the presence of pseudorandom noise. In this case, pseudorandom noise with zero mean and standard deviation of 0.66 mGal was imposed on the anomaly.

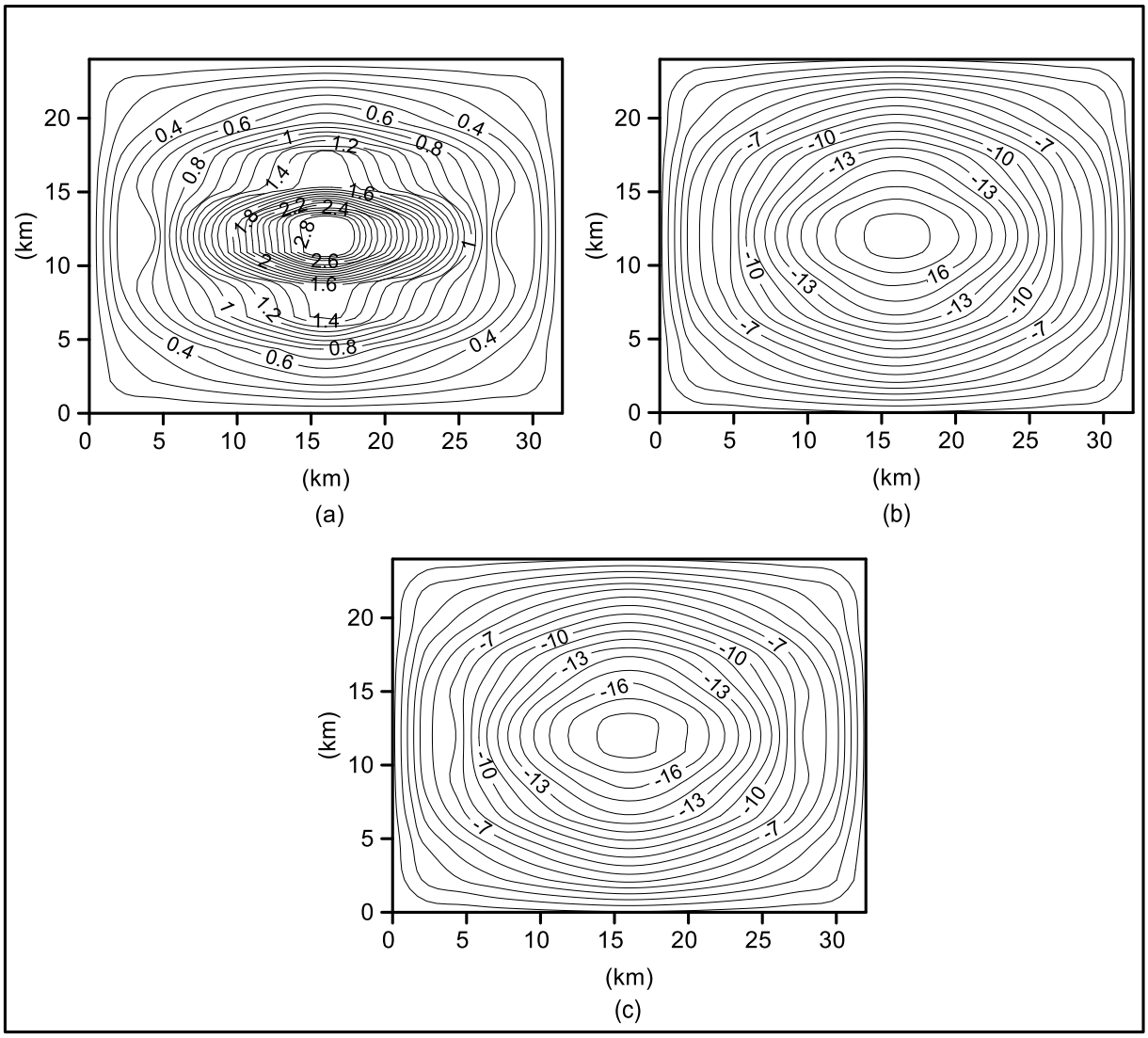


Fig. 4.4 (a) Plan view of a 3D synthetic model of a sedimentary basin, (b) gravity response of the basin with EDM, (c) structure anomaly in the presence of pseudorandom noise.

Considering the noisy anomaly (Fig. 4.4c) as the observed one, the technique was applied again to examine whether the algorithm could recover the assumed structure or not. In this case, the algorithm had performed 20 iterations, beyond which the resulting misfit fell below the predefined threshold of 0.001 mGal, thereby, the algorithm got terminated. The initial misfit of 0.9 mGal for the starting model was drastically reduced to 0.048 mGal at the end of the 8th iteration (Fig. 4.5). The exponential decay in the misfit within the first few iterations is a testimony to rapid convergence of the solution. The interpreted anomaly and the corresponding estimated structure are shown in Figs. 4.6a and 4.6b, respectively. The difference between the observed noisy anomaly and modeled anomaly after the 20th iteration is shown in Fig. 4.6c;

whereas the differences in the assumed and estimated depths are shown in Fig. 4.6d respectively.

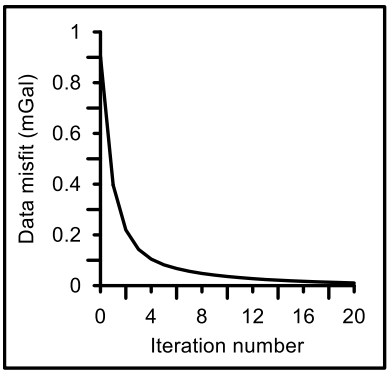


Fig. 4.5 Convergence of the data misfit (mGal) with iteration, synthetic model (Mallesh et al., 2019).

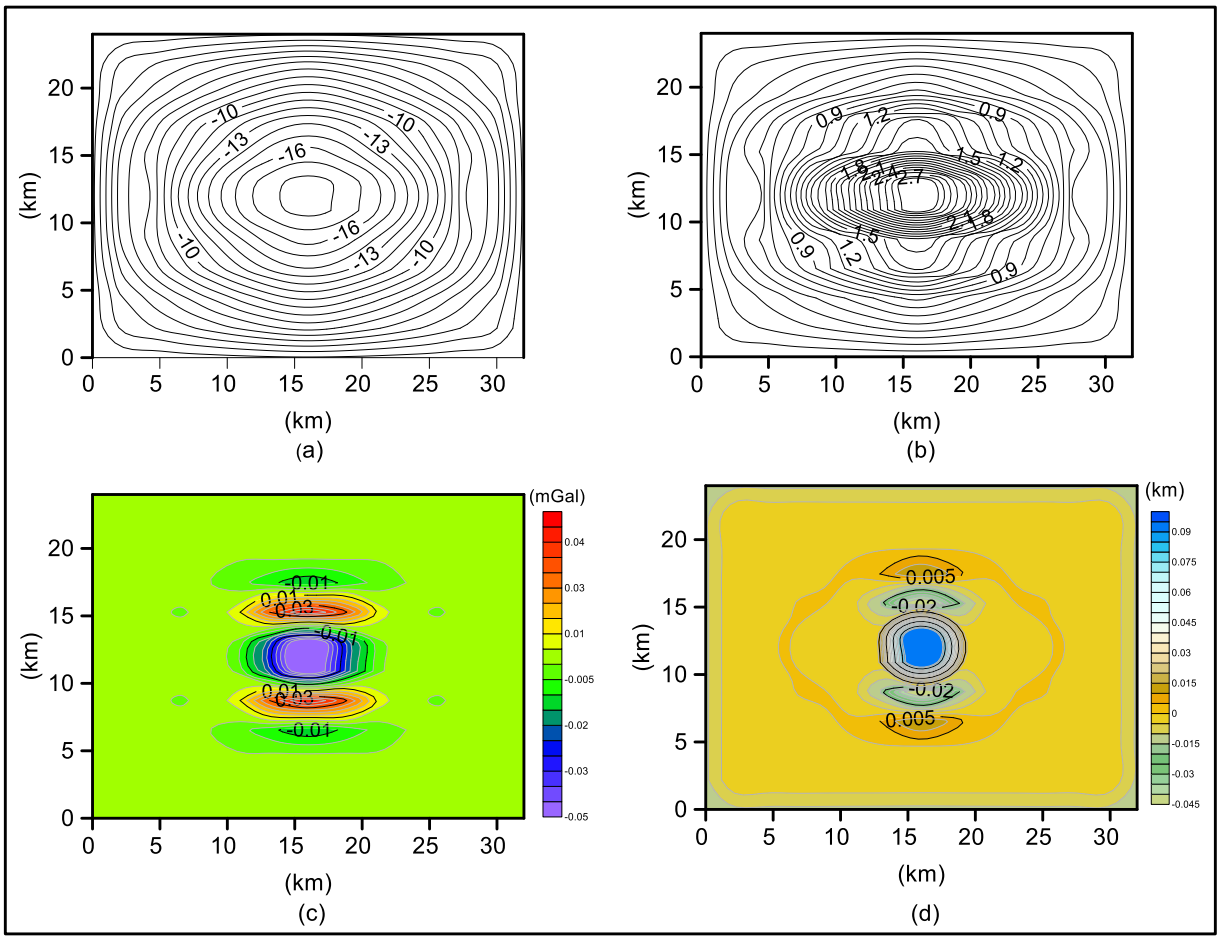


Fig. 4.6(a) Modeled gravity anomaly, (b) estimated structure, (c)residuals between the observed and modeled gravity anomalies,(d)differences between the assumed and estimated depths, synthetic example (Mallesh et al., 2019).

By and large, the modeled anomaly (Fig. 4.6a) closely resembles the observed noisy anomaly (Fig. 4.4c), although a few trivial deviations exist in the interpreted anomaly near the depocentre (Fig. 4.6c). The residuals between the observed and modeled anomalies are within the range of ±0.14 mGal (Fig. 4.6c). Although, the structure features of the dipping basement were successfully recovered by the algorithm (Fig. 4.6b and Fig. 4.4a), the predicted depths show marginal deviations from the assumed one near the depocentre (Figs. 4.4a, 4.6b and 4.6d). For example, the predicted depths to the basement are slightly overestimated by about 2.5% to the north and south of the depocentre, whereas at the centre they are underestimated by 3%. The dimensions of the basement zone at the depocentre were undermined by 19% along the east-west and by 8% along north-south (Fig. 4.4a and Fig. 4.6b). However, the minor deviations in the estimated structure can be treated as under tolerable limits considering the fact that the anomalies used in modeling are noisy.

4.4.2 Field example

The Los Angeles basin in California is known for its great structural relief, complex geologic setting, besides its hydrocarbon potential. The basin was evolved in five different phases, with each one represented by distinct rock assemblage (Yerkes et al., 1965). Based on the structure and rock types, the basin is distinguished into four major blocks, namely, southwestern, northwestern, central and northeastern (Fig. 4.7). Further, each block is delimited by either a major zone of faulting or of flexure in the basement rock (Yerkes et al., 1965). The noteworthy feature of the central block is the presence of a northwest-trending doubly plunging synclinal trough, which contains thick-sectioned sediments.

The major structural features (Yerkes et al. 1965) with residual gravity anomalies (Chai and Hinze, 1988) draped over the digital elevation model (90 m resolution) of the basin (CGIAR-CSI GeoPortal at http://srtm.csi.cgiar.org/) is shown in Fig. 4.7. One can clearly notice from Fig. 4.7 that all the major structural features of the basin are well brought out by the anomaly map. For example, the boundary faults of the central block are well correlated with the steep gradients in the anomaly. Similarly, the doubly plunging synclinal trough is manifested by a well-defined negative anomaly. The doming-up of contours in the southwestern block and

down-warping of contours in the western part of the central block are well correlated with the mapped anticlinal structures.

Chai and Hinze (1988) have analyzed the gravity anomalies of the basin for its basement structure (Fig. 4.10b) by assigning a prescribed EDM to the basin fill defined by

$$\Delta\rho(z) = -0.5e^{-0.1609z},$$

which was obtained from McCulloh's (1960) drill-hole sample density data.

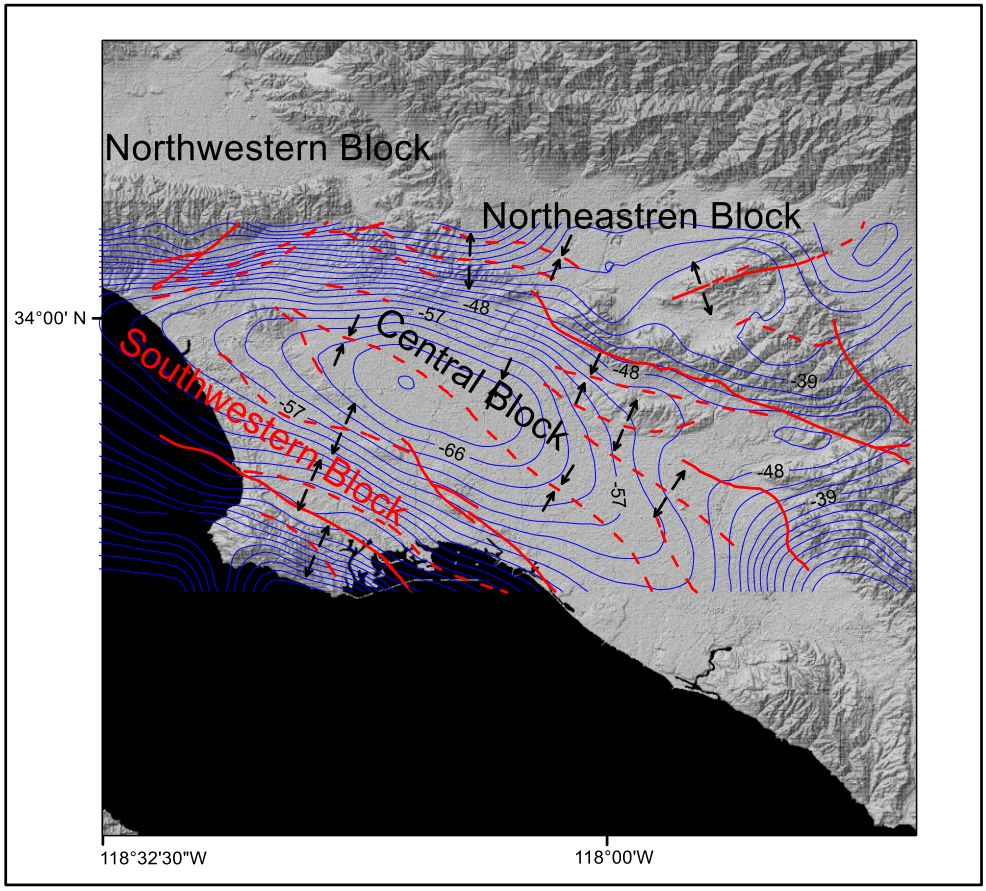


Fig. 4.7 Observed gravity anomalies (Chai and Hinze 1988), mapped structural features (Yerkes et al. 1965), and digital elevation model of the Los Angeles Basin, California. Solid lines in blue are the observed gravity anomalies, solid lines in red represent faults or fault zones (dashed where approximately located), solid lines in black with opposite arrow heads indicate anticlines, solid lines in black with arrow heads inwards indicate syncline (after Mallesh et al., 2019).

For the present modeling, the gravity anomaly map (Chai and Hinze 1988) was digitized into 375 nodes (Mallesh et al., 2019), sampled at an interval of 3.16 km along the x-axis and 2.82

km along the *y*-axis (Fig. 4.8a). The data was interpreted by specifying the allowable data misfit between the observed and modelled anomalies as 0.01 mGal. The algorithm had performed 157 iterations, after which the resulting misfit had attained a value larger than its previous one, thereby forced the algorithm for its termination.

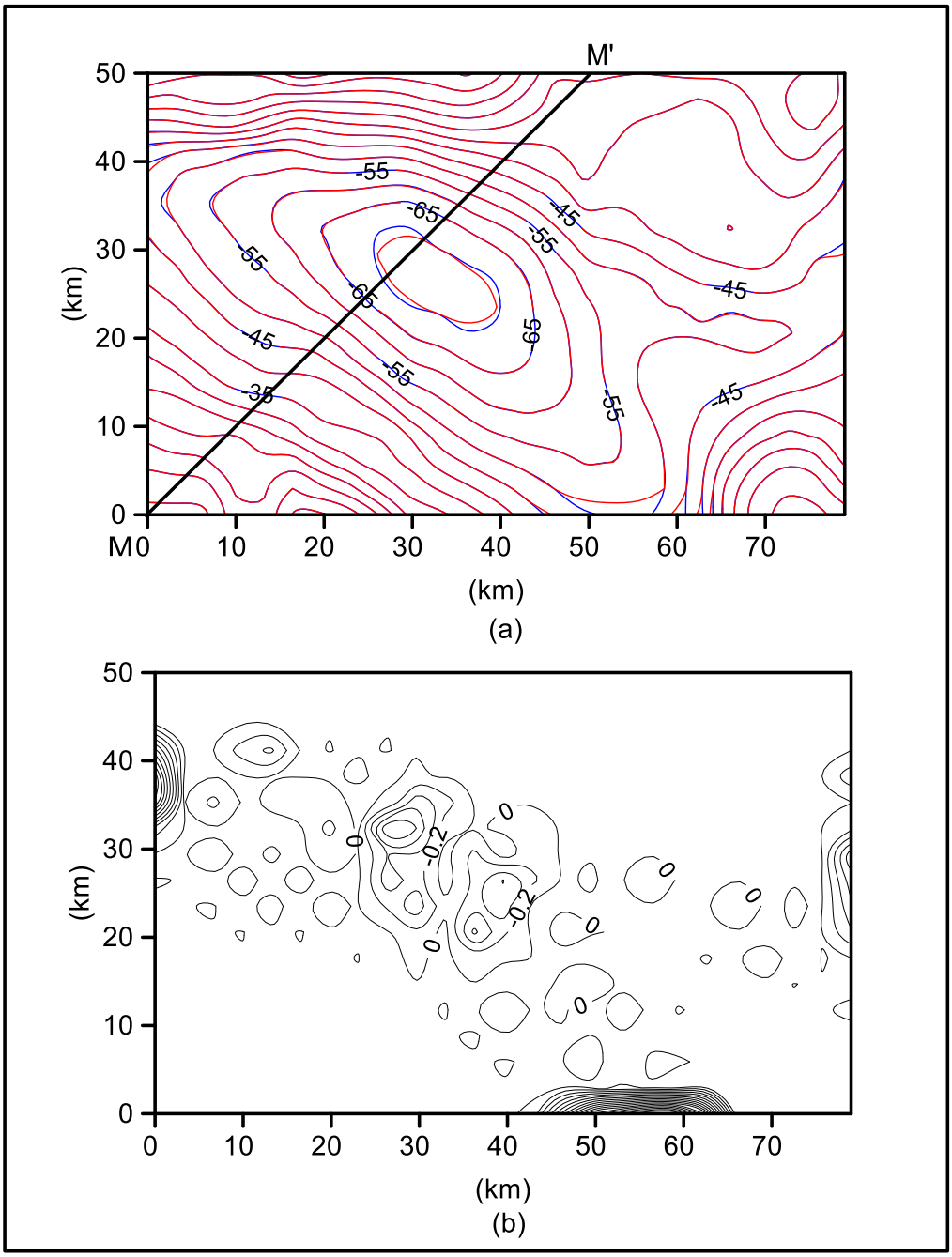


Fig. 4.8 (a) Observed (solid lines in blue) and modeled gravity anomalies (solid lines in red), (b) residuals between the observed and modeled gravity anomalies, Los Angeles basin, California (after Mallesh et al., 2019).

Overall, the modelled anomaly after the 157th iteration shown as solid line in red in Fig. 4.8a closely resembles the observed anomaly (solid line in blue in Fig. 4.8a). A few minor deviations are observed between the two anomalies at the centre of the basin and on the southern boundary. Fig. 4.8b shows the residuals between the observed and modelled anomalies after the 157th iteration, being the concluding one. One can notice that majority of the misfits across the model space is near zero (Fig. 4.8b). The misfit, which has attained its maximum value of 5.08 mGal for the starting model has reduced drastically to 0.45 mGal at the end of the 14th iteration (Fig. 4.9), beyond which the decay was rather gradual and smooth. The estimated depth structure of the basin from present modeling is shown in Fig. 4.10a with the surface structural features (Yerkes et al., 1965) superimposed. It is clearly noticed from Fig. 4.10a that all the structural features reported by Yerkes et al. (1965) are well reflected on the estimated model too. The structure of the basin inferred by Chai and Hinze (1988) is shown in Fig. 4.10b for comparison.

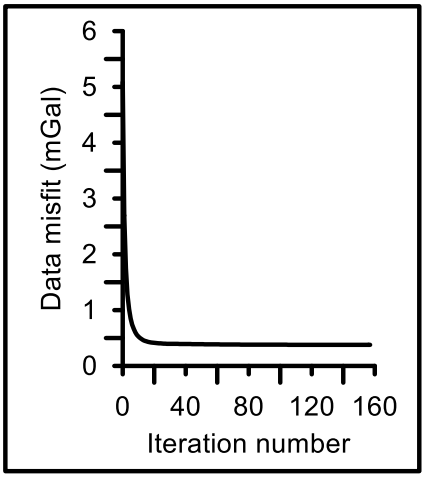


Fig. 4.9 Variation of data misfit against iteration, Los Angeles basin, California

The maximum depth to the floor of the basin obtained from the present method is 9.73 km (Fig. 4.10a), whereas Chai and Hinze (Fig. 4.10b) reported 10 km. The structure contour map prepared by McCulloh (1960) on the basis of surface geology, drill-hole and seismic reflection data also had revealed a figure of 9.5 km for the maximum thickness of the basin. By and large, the modeled structure deciphered from the present method is comparable with the one reported by Chai and Hinze (1988), however, a few noticeable exceptions are evident. For example, the estimated structure from the present method has revealed the presence of two north-south trending basement depressions (confined between $x \in$ (25 km, 45 km) and $y \in$ (20 km, 35 km))

separated by a basement high. Such a moat-like structure in the basement was not reflected prominently in the model of Chai and Hinze (1988).

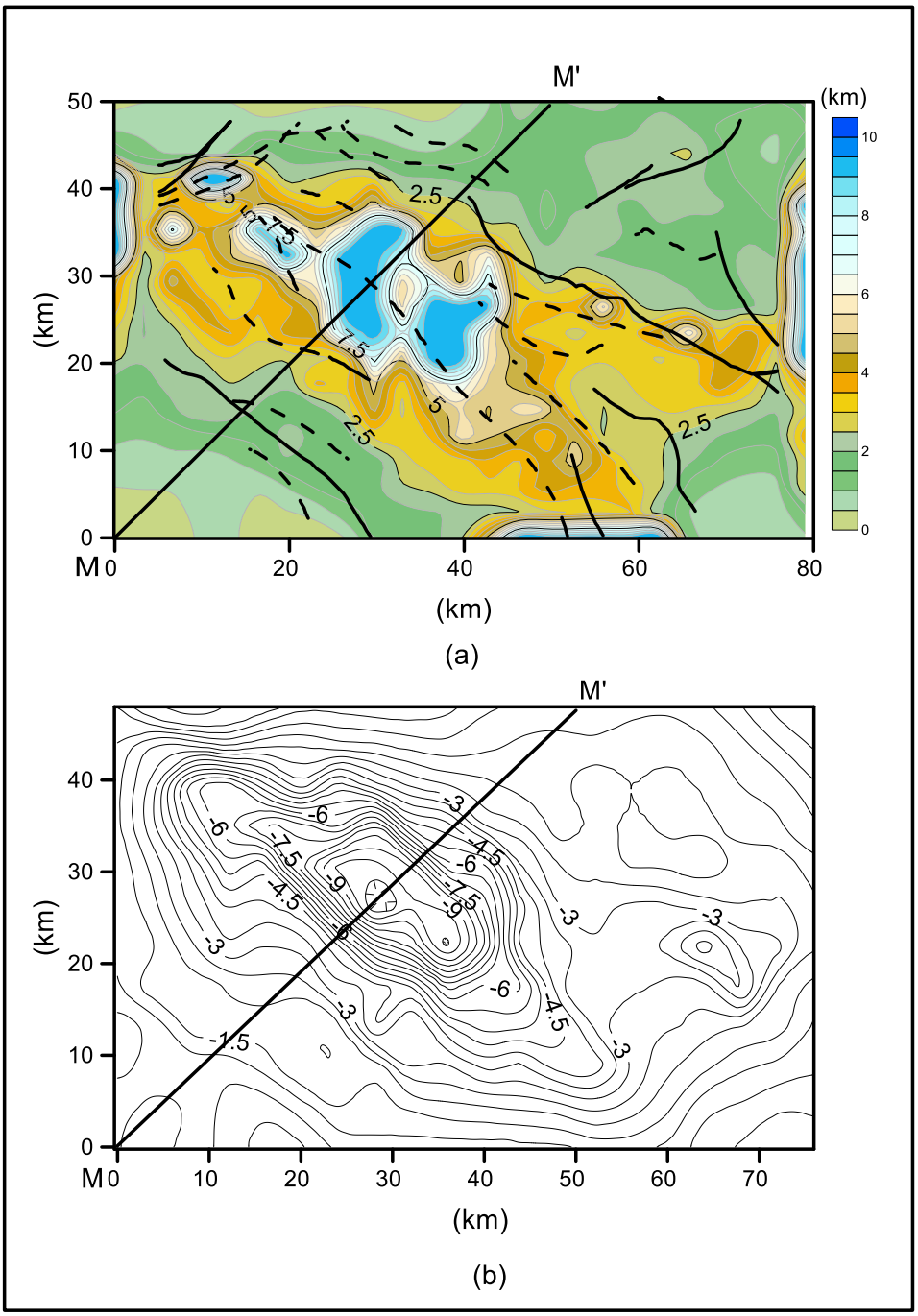


Fig. 4.10(a) Estimated structure of the Los Angeles basin, California from present method. The structural features mapped by Yerkes et al. (1965) are also shown, (b) inferred structure of the basin by Chai and Hinze (1988). Note that the depth contours are drawn at 0.5 km interval. MM' is a selective profile along which the cross-sections of the basin from (a) and (b) are shown in Fig. 4.11.

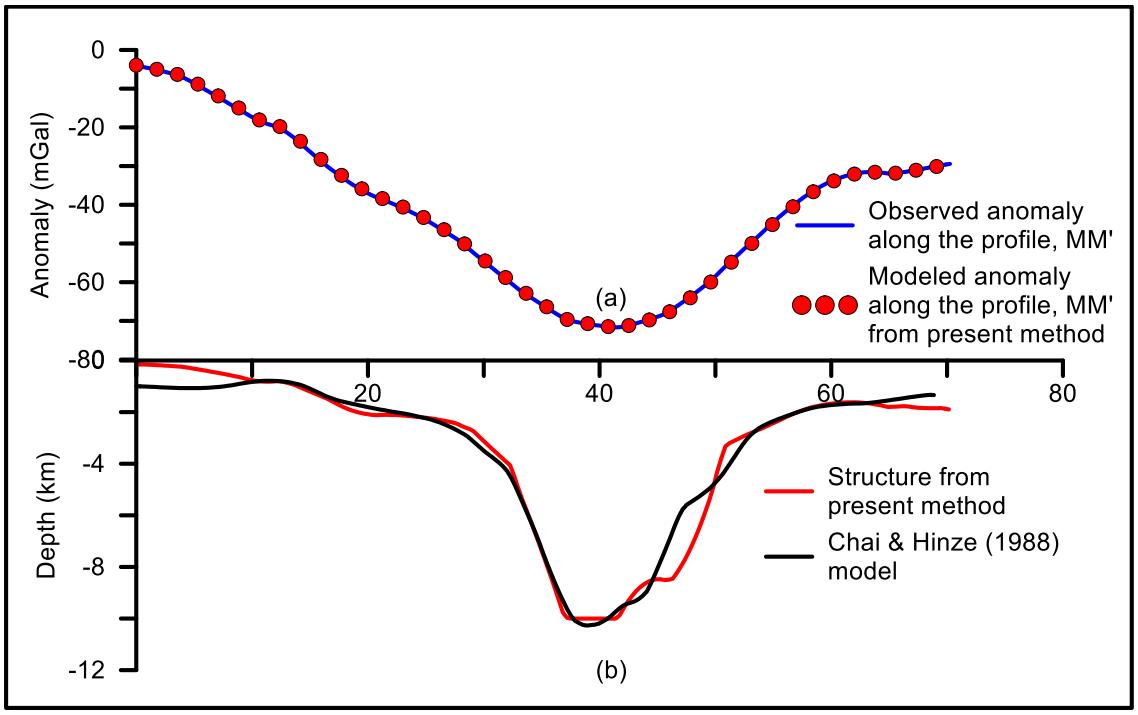


Fig. 4.11(a) Observed and modeled gravity anomalies along the Profile, MM', from present method (b) estimated depth structures of the basin from present modeling and the one by Chai and Hinze (1988).

Fig. 4.11a shows the observed and modelled gravity anomalies of the basin along a selective profile, MM' derived from Fig. 4.8a. The estimated structure reliefs of the basin along the same profile obtained from the two interpreted depth models (Figs. 4.10a and b) are shown in Fig. 4.11b. Both cross-sections divulge steeply dipping fault system on the south-western side of the basin beyond 4 km depth, whereas on the north-eastern side the basin is delimited by a listric fault. The present interpreted model indicates in line with the structural map of Yerkes et al. (1965) that the sedimentary basin appears to extend beyond the southern boundary of the interpreted grid. Such an inference was missing in the model deciphered by Chai and Hinze (1988).

4.5 Results and discussion

i) A space domain based 3D algorithm is presented along with a software, POLYMOD3D, to infer sedimentary basin structure from spatially distributed observed gravity anomalies, with EDM describes the density contrast with depth.

ii) In contrast to the method of Talwani and Ewing (1960), the present modeling technique approximates the 3D anomalous volume by an ensemble of multiple vertical laminas of polygonal shape, each one with unit thickness. Such simulation of model geometry has a prominent advantage over the method of Talwani and Ewing (1960) in the sense that it is relatively easy to design interpretational strategies to model subsurface geologic structures from the gravity anomalies.

iii) Owing to the fact that no closed from analytical solutions could be derivable in the spatial domain using EDM, a method that combines both analytic and numeric approaches is presented to realize forward modelling.

iv) The proposed technique calculates the initial depths of a basement structure from the measured gravity anomalies and afterwards improves the depth estimates in an iterative approach till one of the prescribed termination criteria satisfies.

v) Two examples, one synthetic and a real, are presented to demonstrate the applicability of the technique. In case of the synthetic example, gravity anomalies produced by a known but complex structure in the presence of pseudorandom noise are analysed. The modeling algorithm has yielded a structure that is, on the whole, congruent to the assumed structure even in the presence of pseudorandom noise.

vi) In real field example, the anomalies resulted from the Los Angeles basin, California are analysed. Chai and Hinze (1988) have originally interpreted the anomalies of the basin making use of a derived EDM. In the present study, the anomalies are analysed using the same density-depth model. The estimated depths of the basin are in close agreement with those inferred by Chai and Hinze (1988) and Yerkes et al (1965). A few deviations noticed between the present estimated structure and the one by Chai and Hinze (1988) are explained.

vii) The present method is simple, efficient and effective in its implementation when compared to existing 3D interpretation methods that make use of the exponential density model. The proposed technique has general applicability to basin delimitation.

CHAPTER
FIVE

3D Spatial domain gravity inversion with multiple polygonal cross-sections and EDM *

5.1 General

Gravity method plays a decisive role in deciphering structural configurations of buried density interfaces. Distribution of sediments within a basin is largely controlled by the nature of basement topography during the process of sedimentation and also during post depositional tectonic adjustments. Hence, quantifying basement configurations from surface geophysical measurements is always pave the way to build efficient strategies for resource exploration. Although 2D and 2.5D techniques are available for quantification of anomalies these schemes yield subsurface information along selected profiles only. Though 3D strategies are computationally expensive (compared to 2D and 2.5D) these schemes undoubtedly yield in-depth information on the spatial distribution of subsurface density.

Cordell and Henderson (1968) had developed a 3D method to analyze gravity anomalies, wherein, the subsurface anomalous mass was described by several vertical prismatic structures; and vertical position of each prism was scaled with reference to a horizontal plane at depth. To speed up the convergence of solution, they had suggested the use of anomaly expression of a vertical cylinder to calculate approximate gravity effect of a prism on the top of its axis, whereas line-source formula to calculate the gravity effects at remaining nodes. Gerard and Debeglia (1975), "Murthy et al. (1990) and Rao et al. (1999)" also had followed more or less similar approach to analyze the gravity anomalies produced by density interfaces. The effectiveness of the method proposed by Gerard and Debeglia (1975) is largely reliant on the spectral content of the anomalies. On the other hand, source depths estimated by conventional Euler deconvolution

---

* *In press with Journal of Earth System Science.*

method (Thompson, 1982; Hansen and Suciu, 2002; Toushmalani and Hemati, 2013) are prone to yield large errors particularly when the anomalies are noisy (Luo et al., 2018). Kilty (1983) and Murthy et al. (2000) have demonstrated that basement depths estimated from potential field anomalies using Werner deconvolution technique (Werner, 1953; Hortman et al., 1971) often show large vertical scatter, making the interpretation more often complicated. Presuming that seismic impedance contrast correlates equivalently well with the bulk density contrast, Panzner et al. (2011) had proposed a 3D technique that involves the use of prior subsurface information obtained from seismic stereotomography to analyze the gravity anomalies. However, in reality rocks that produce impedance contrasts need not necessarily be associated with changes in bulk density (Avseth et al., 2001; Han and Batzle, 2002). On the other hand, representing the gravity field of a 3D source and its gradients by 3D Cauchy-type integral, Zhdanov and Cai (2013) developed a 3D inversion technique to comprehend density contrast surfaces from gravity anomalies. Nevertheless, all the enlisted strategies presume that sedimentary rocks possess uniform density throughout the basin volume, which is rarity in reality.

Pohanka (1988), and Hansen (1999) presented formulae using linear density function to realize anomaly computations of gravity fields of polyhedral bodies. Holstein (2003) had presented gravimagnetic anomaly formulas for polyhedra of spatially linear media, and "Hamayun et al. (2009)" obtained an expression for gravity potential of a polyhedral source using linear density function. On the other hand, "Rao et al. (1990), Abdeslem (2005), Wu and Chen (2016), Jiang et al. (2017), Zhang and Jiang (2017), Fukushima (2018), and Liu et al. (2019)" have derived expressions for gravity anomalies of prismatic sources considering polynomial functions to simulate density-depth dependence of source rocks. "D'Urso and Trotta (2017), Ren et al. (2017)" also had used polynomial functions to derive analytical gravity expressions of polyhedral bodies, whereas, Jiang et al. (2017) used polynomial density functions to derive the equations for gradient tensor of 3D prismatic bodies. Chen et al. (2018) used polynomial density functions for obtaining gravity anomaly of a polyhedral prism. In recent past, Chakravarthi et al. (2013a), Mallesh et al. (2019) have presented schemes using exponential density model to realize forward modeling of parallelepipeds and 3D polygonal source bodies in the spatial domain.

In this Chapter, an automatic 3D inversion technique and related software is developed to estimate the depths of concealed density contrast surfaces using the measured gravity anomalies. Forward modelling is performed in the spatial domain using a predefined EDM by a technique that combines both analytic and numeric approaches (Mallesh et al., 2019). The applicability of the inversion is exemplified with the synthetic gravity anomalies produced by a model having known geometry in the presence of pseudorandom noise. Later, this technique was utilized to infer the structure of the Almaz$\acute{a}$n basin in northeast Spain using the measured gravity anomalies with derived EDM. In the case of theoretical example the derived information from inversion was compared with the assumed one, and in the case of Almaz$\acute{a}$n basin the result was discussed in the light of the reported information derived from seismic data.

5.2 Forward modeling – Theoretical considerations

Equations (4.5) to (4.9) presented in Chapter-4 under section 4.2 are used in the present inversion to execute forward modeling of 3D sedimentary basins with prescribed EDMs. The source volume is approximated as a collage of vertical unit laminas in the xz plane arranged along the y-axis.

5.3 Optimization of gravity anomalies

Optimization refers to a mathematical exercise of solving optimum depth parameters of a density interface (i.e., depth ordinates of vertices of multiple vertical polygons) using the measured gravity anomalies at plurality of observations on the plane of observation. This could be achieved by fitting the observed gravity anomalies to the anomaly expression (equation (4.5)) following some predefined convergence criteria, so that the theoretical gravity anomalies closely resemble the measured ones. As in the case of automatic modeling the basin relief is treated as zero over and beyond the basin boundary, thereby the problem of optimization becomes to solve $(N_x - 2)^* (N_y - 2)$ unknown parameters (depth ordinates) from $N_x{*}N_y$ anomaly values, where N_x and N_y represent the number of measurements along the x and y −axes respectively. To start with, approximate depths to a density interface are calculated (excluding the ones along the periphery) using equation (4.10). Equation (4.5) determines the gravity result of the initial structure, $\Delta g_M(x_{i'}, y_{j'})$, i'=1, 2,…,N_x and j'=1, 2,…,N_y. The

discrepancy between the measured ($\Delta g(x_{i'}, y_{j'})$), and theoretical ($\Delta g_M(x_{i'}, y_{j'})$) anomalies at a station, $(x_{i'}, y_{j'})$, can be formulated as (Rao et al., 1999, Chakravarthi, 2003)

$$\Delta g(x_{i'}, y_{j'}) - \Delta g_M(x_{i'}, y_{j'}) = \sum_{i=2}^{N_x-1} \sum_{j=2}^{N_y-1} \frac{\Delta g_M(x_i, y_j)}{\partial z} dz(x_i, y_j). \quad (5.1)$$

Linear equation akin to equation (5.1) is set up for each observation, $(x_{i'}, y_{j'})$, i'=1, 2,…,N_x and j'=1, 2,…,N_y. Normal equations, $(N_x - 2)$*$(N_y - 2)$, in number are then framed from the set of linear equations and solved for the depth improvements, $dz(x_i, y_j)$. Here, i varies from $2,..,(N_x - 1)$, and j from $2,..,(N_y - 1)$ respectively. The Ridge Regression algorithm (Marquardt, 1970) is used to solve the unknowns, $dz(x_i, y_j)$, by minimizing the data misfit given by equation (4.11) (Mallesh et al., 2019).

The normal equations to be solved are given by Ramamma et al. (2020) as

$$\sum_{j'=1}^{N_y} \sum_{i'=1}^{N_x} \sum_{m=1}^{(N_x-2)*(N_y-2)} \frac{\Delta g_M(x_{i'}, y_{j'})}{\partial a_n} \frac{\Delta g_M(x_{i'}, y_{j'})}{\partial a_m} (1 + \delta\vartheta) da_m$$
$$= \sum_{j'=1}^{N_y} \sum_{i'=1}^{N_x} \frac{\Delta g_M(x_{i'}, y_{j'})}{\partial a_n} [\Delta g(x_{i'}, y_{j'}) - \Delta g_M(x_{i'}, y_{j'}], \quad (5.2)$$

Here, $n = 1,2,,\ldots..,(N_x - 2) * (N_y - 2)$. Also, da_m represents the improvements in the depth ordinates of the vertices of multiple polygons. ϑ is assigned a value of 1 for $i' = j'$, otherwise to zero. δ is the damping factor, whose magnitude is controlled by the algorithm depending upon whether the existing data misfit is more than or less than its preceding value (Chakravarthi, 2003; Ramamma et al., 2020). The existing depth ordinates of multiple polygons are updated, iteratively, by adding/subtracting the depth improvements, da_m, estimated from equation (5.2) till one of termination criteria satisfies as detailed in section 3.3.

5.3.1 POLYINV3D

Following the principles of inversion and the methodology described in the text, a software named POLYINV3D, coded in Core Java was developed and presented in Annexure 5-A to find

depths of 3D density boundaries from spatially distributed gravity data set using a prescribed EDM. The software follows Model-View-Controller architecture (Fig. 2.5).

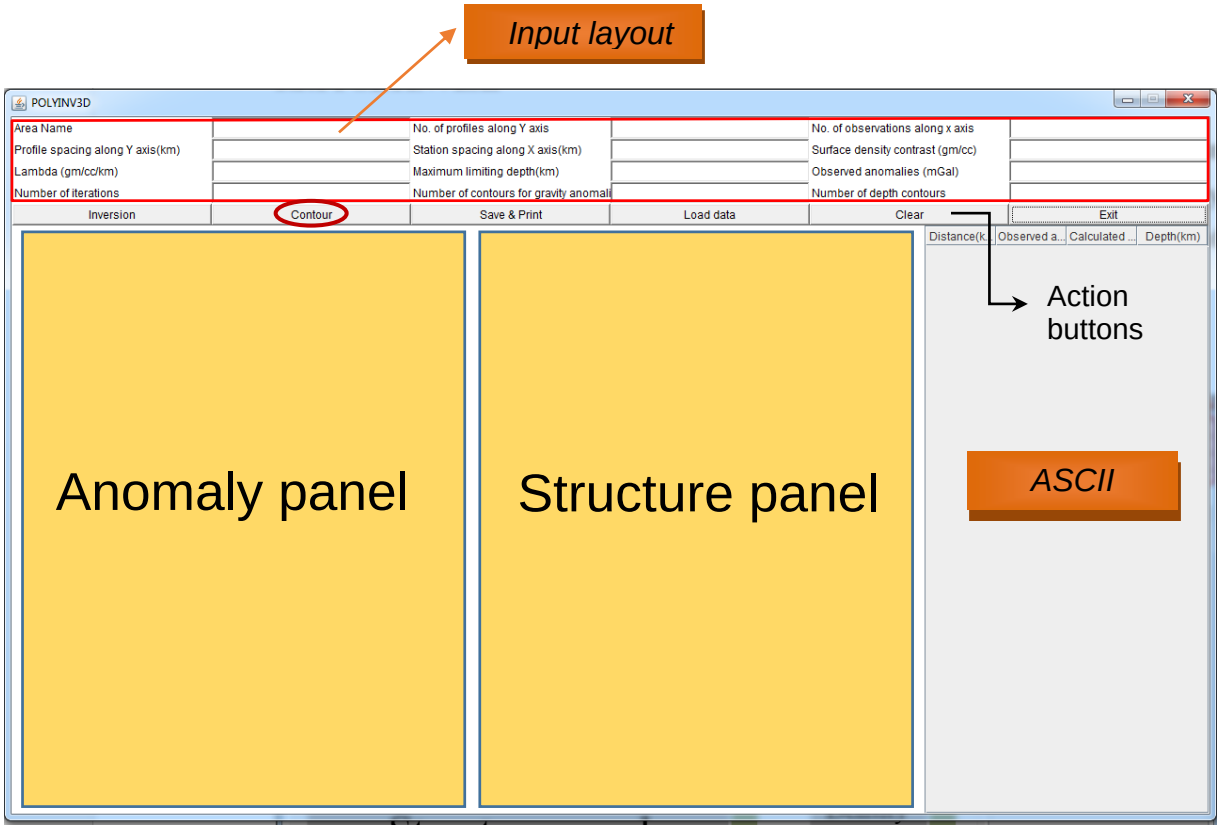


Fig.5.1 View module of POLYINV3D

Fig. 5.1 depicts the view module of POLYINV3D, which appears upon the activation of the batch file. Inputting of data to the software can be realized by either i) entering the data in a formatted excel sheet and then read the data file by 'Load data' action button, or ii) using the fields of the input layout (Fig. 5.3). The input data required to run the software are Area name, quantity of profiles considered along the y-axis, number of observations each profile consists, spacing of profiles along the y-axis (km), spacing of stations along the x-axis (km), density contrast observed at the surface (g/cm^3), constant of EDM (g/cm^3/km), maximum limiting depth (km), measured gravity anomalies covered sequentially from the first profile onwards (mGal), number of iterations, number of contours required for plotting of observed and modelled gravity anomalies, and the number of contours required for plotting basement geometry. Upon invoking the action button 'Inversion' (located below the input layout panel) the software performs the inversion within the prescribed convergence criteria in iterative mode. As in the case of

POLYMOD3D, the graphical layout displays the measured and theoretical gravity anomalies, and inferred basement configuration in respective contour maps. The improvements in model space and corresponding modelled gravity response are also displayed in animated versions against the iteration. Finally, the ASCII layout displays the out results in a tabular form, profile-wise. The user opts for 'Save & Print' action button to save the output file to the destination followed by printing.

5.4 Applications

Two residual gravity anomaly maps, one attributed to a theoretical model and the other one related to the Almaz*á*n basin in northeast Spain, are analyzed by the proposed inversion technique to demonstrate its applicability. In either case, the density contrast of sedimentary rocks varies as a function of depth following equation (1.12). During the iterative process of inversion, constants of the prescribed EDM are kept unchanged and allowed the depth parameters of the model space to improve until one of the termination criteria is achieved. In the case of theoretical model, the technique is applied both with and without pseudorandom noise.

5.4.1 Theoretical example

Fig. 5.2a shows a theoretical model of north-south striking basin structure whose spatial dimensions are represented by 18 km X 27 km. The structure of the basin is described with depth contours that are scaled along the z-axis (Fig. 5.2a). The model space is reflected with several inward dipping fault systems towards the depocentre from the margins of the structure. The fault systems show steep dips up to a depth of 2.8 km, beyond which the dips are moderate. A well-defined basement depression (depocentre) is present in the north central part of the basin (Fig. 5.2a), where the sediment thickness attains its maximum (3.42 km).

Presuming that the sediment density within the basin obey exponential decrease with depth ($\Delta\rho_0$= -0.451 g/cm^3 and λ = 0.4211 g/cm^3/km as shown in Fig. 5.2b), the structure anomaly was calculated using equation (4.5) and shown in Fig. 5.2c. The magnitude of gravity anomaly varies from -1 mGal to -28 mGal within the structure. It is clearly evident from Fig. 5.2c that the gravity low of -27 mGal is not confined to the depocentre alone but it extends over a larger part

of the basin along its strike, where the thickness of sediments is of significance. It can be seen from Figs. 5.2a and c that all boundary faults of the structure up to 2 km depth are well reflected on the anomaly map, but not the structures beyond 2 km depth. The density contrast of the section of sediments from 3.2 km depth and beyond shows more or less similar magnitudes, hence, independent gravity lows corresponding to the depocentre were not reflected prominently on the anomaly map.

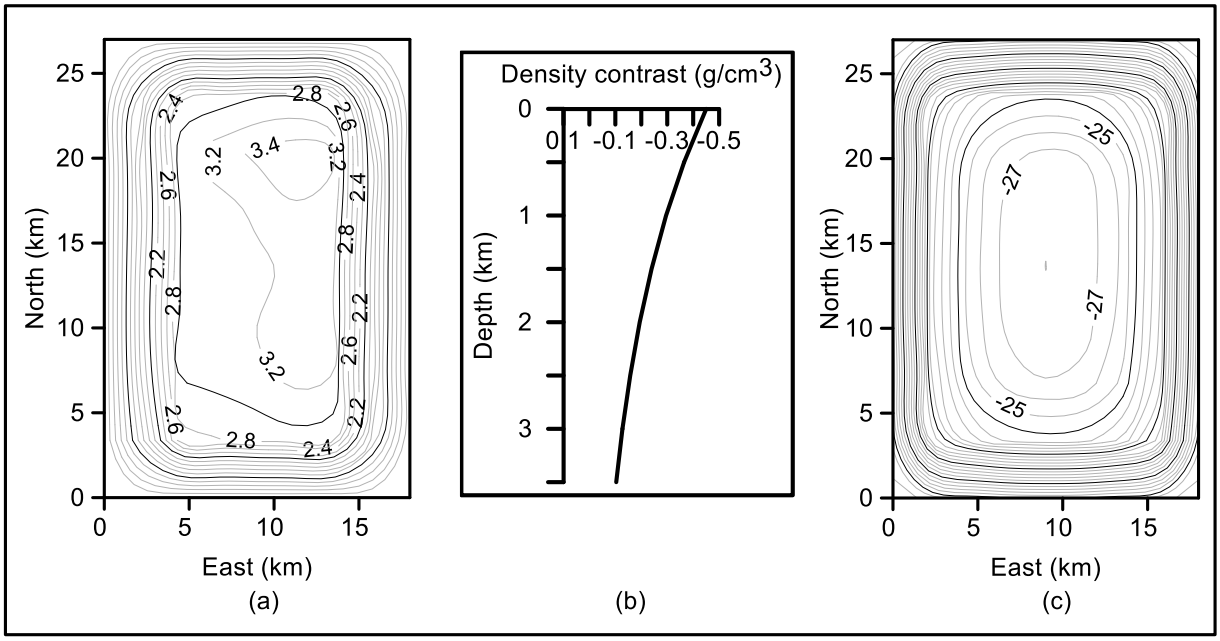


Fig. 5.2(a) Theoretical model of a typical intracratonic sedimentary basin, (b) prescribed EDM, and (c) theoretical gravity response of the structure.

Assuming that the theoretical anomaly shown in Fig. 5.2c as the observed one, the inversion technique was applied to examine whether or not it could restore the basement structure. The inversion algorithm had run 5 iterations, beyond which the misfit had attained a value less than the predefined threshold. The damping factor, δ, as well as the misfit, J, have decayed consistently with the iteration number, more sharply in the first 3 iterations and thence gradually till the end of the 5th iteration. The theoretical gravity response as well as the estimated structure after the 5th iteration exactly replicate the anomaly shown in Fig. 5c and the structure in Fig. 5a (for brevity this case was not shown).

In the second phase, the efficacy of the technique was tested by adding random noise to the structure anomaly (Fig. 5.2c) before the inversion was attempted. Fig. 5.3a shows the noisy

anomaly of the structure. The standard deviation of pseudorandom noise in this case was 0.87 mGal,, which is significant because a value of 0.5 mGal is usually treated as considerable in gravity inversion problems (Barbosa, 1999; Chakravarthi and Sundararajan, 2007b).

When the inversion was performed on the noisy data (Fig. 5.3a) the algorithm took 11 iterations, after which the damping factor had attained a large value thereby forced the algorithm for its termination. The initial data misfit (equation (4.11)), for the initial model space was 2.994 mGal, which drops down drastically to 0.1248 mGal after the 10^{th} iteration. Beyond the 10^{th} iteration, the data misfit had attained a value of 0.169016 mGal resulting in the formation of a ridge in the misfit curve (shown as solid line in Fig. 5.3c). At this juncture, the existing damping factor, δ, was doubled to a magnitude of 0.00097 and the system of equation (5.2) was solved again for the depth improvements. Using the updated depth parameters, theoretical anomalies of the model space were recalculated for which a new misfit (0.169036) was obtained. Yet again, the magnitude of new data misfit was found more than the one obtained after the 10^{th} iteration. Hence, the above procedure of doubling the damping factor, solving normal equations, updating of depth estimates of the model space was repeated as many as thirteen times within the inner loop of the 11^{th} iteration , beyond which the resulting misfit had attained a magnitude (0.1112 mGal) lesser than the misfit realized at the end of the 10^{th} iteration (0.1248 mGal). Fig. 5.3c shows graphically how the data misfit and corresponding damping factor have changed within the inner loop of the 11^{th} iteration. The overall changes in the damping factor and data misfit against the iteration are shown in Fig. 5.3d.

The recalculated gravity response of the structure after the 10^{th} iteration (being the concluding one) was shown in Fig. 5.3b and the corresponding inferred structure in Fig. 5.3e respectively. On the whole, the nature of fit between the observed noisy anomaly and that of the recalculated anomaly after the inversion was found satisfactory, however, a few deviations persists. For e.g., the spatial dimensions of gravity lows (-24 mGal to -28 mGal contours) in the modeled anomaly (Fig. 5.3b) are small when compared to the observed noisy anomaly (Fig. 5.3a). The estimated structure shown in Fig. 5.3e compares fairly well with the assumed structure (Fig. 5.2a), albeit a couple of deviations exist. All the characteristics of boundary faults of the basement are well resolved and restored in the estimated structure up to a depth of 2.8 km.

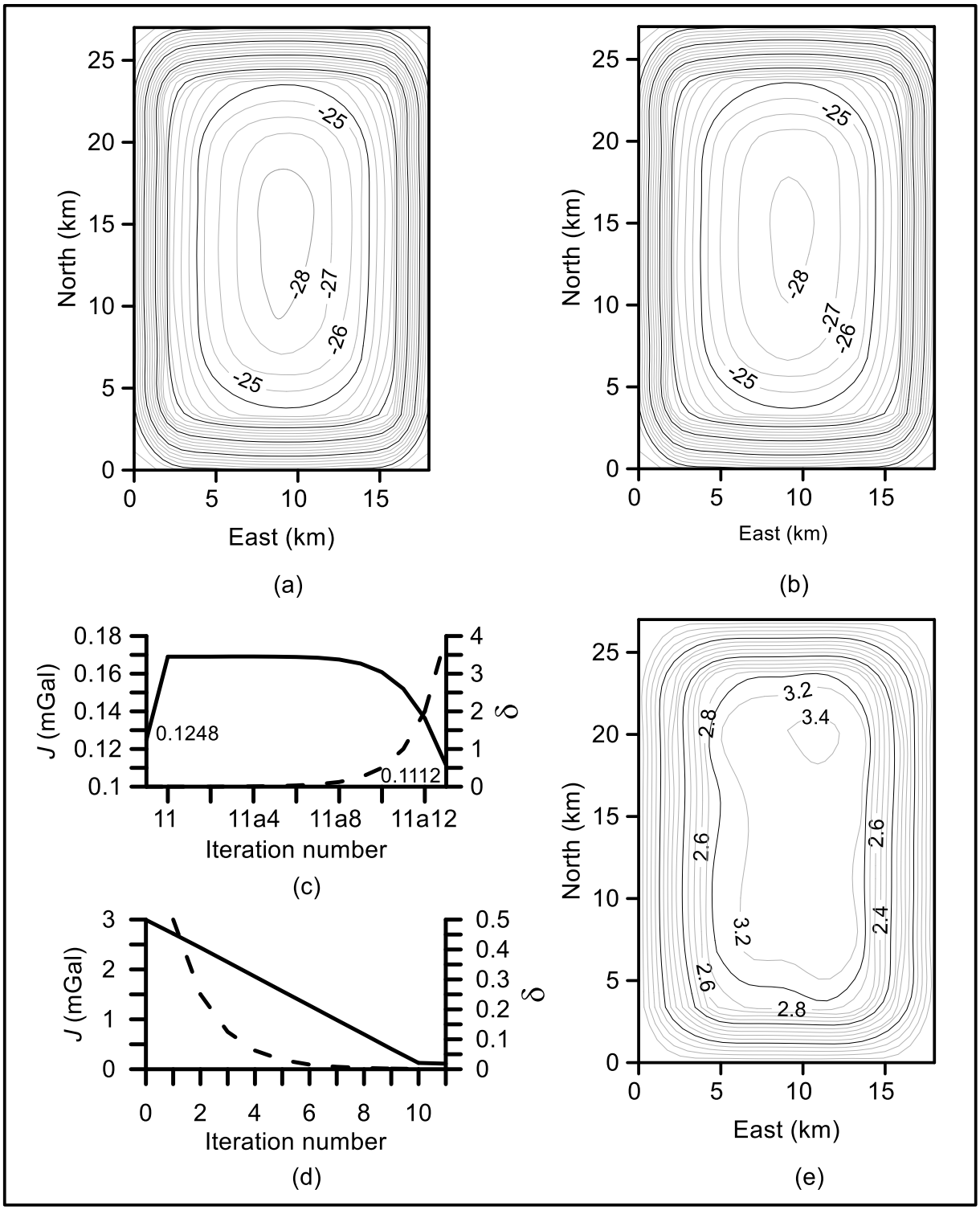


Fig. 5.3(a) Noisy gravity anomalies of the structure, (b) recalculated gravity response at the end of the 10th iteration, (c) variation of misfit (solid line) and damping factor (dashed line) within the inner loop of 11th iteration, (d) overall changes in the misfit (solid line) and damping factor (dashed line) with iteration number, (e) recovered depth structure of the basin after the 10th iteration.

The geometry of the southward nosing of the basement observed in the assumed structure (represented with 3.2 km depth contour) (Fig. 5.2a) was not fairly recovered in the inferred structure (Fig. 5.3e). The maximum depth estimated from present inversion (3.44 km) is in line with the assumed maximum depth (3.42 km), however, the dimensions of the estimated depocentre were moderately underrated (Fig. 5.3e). These differences between the assumed and estimated structure are fairly tolerable in view of the fact that the anomalies used in the inversion are noisy.

5.4.2 Field example

The Almaz*á*n Basin, located about 200 km Northeast of Madrid, was formed between two well-known mountain ranges, namely, the Iberian Range towards the east and the Central System towards the southwest in the Northeast Spain (Fig. 5.4). It forms the southeast extension of the Duero Basin. The basin evolution was attributed to the reactivation of northeast-southwest and northwest-southeast oriented basement faults triggered by northeast-southwest and north-south compression within the intracontinental structural regimes (Bond, 1996).

The basin is more or less crescent shaped with a distinct northwest-southeasterly trend covering approximately 4200 sq. km area. Along the north-eastern margin of the basin Eo-Oligocene sediments are exposed, and in the central and southern parts Miocene sediments present. The striking feature of the basin is the absence of Late Jurassic and Early Cretaceous deposits from central and western zones of the basin. A complete picture on the evolution of the basin and its tectonics was described by Bond (1996).

G*ó*mez-Ortiz *et al.* (2005) have modeled the regional gravity anomalies of Central Spain covering the Duero Basin, Almaz*á*n Basin, Tajo Basin and Spanish Central System to decipher deep crustal density structure. In order to distinguish gravity signatures between shallower and deeper sources that have overlapped wavelengths, the gravity anomalies attributed to the Cenezoic sedimentary fill was computed and subsequently removed from the observed gravity anomalies before modeling was performed to decipher deep crustal structure.

The measured gravity anomaly of the Almaz*á*n Basin (G*ó*mez-Ortiz *et al.*, 2005) shown in Fig. 5.5 was arrayed over the DEM (Digital Elevation Model) of the basin and its surroundings

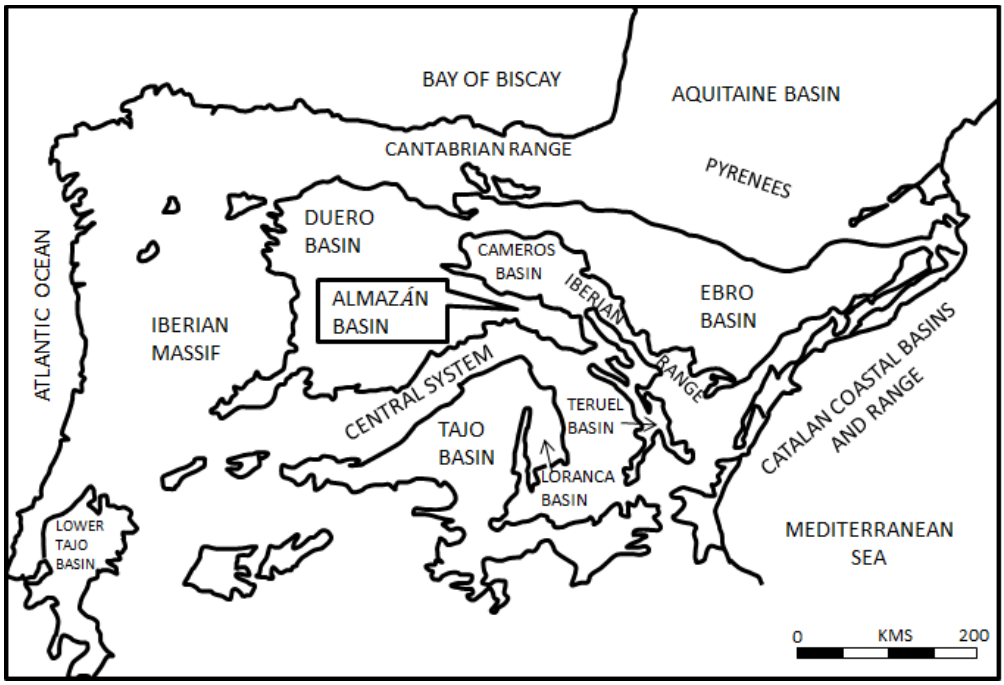


Fig. 5.4 Location map of the Almazán Basin, northeast Spain and its surroundings (modified after Bond, 1996).

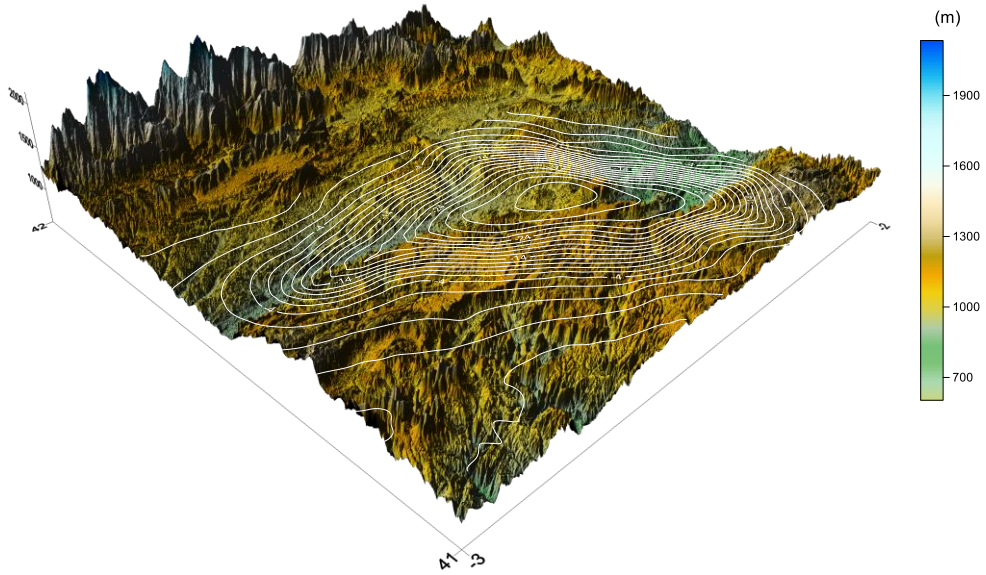


Fig. 5.5 Gravity anomalies draped over the digital elevation model at 90 m resolution, Almazán Basin, northeast Spain (Ramamma et al., 2020).

at 90m spatial resolution "(CGIAR-CSI GeoPortal at http:// srtm.csi.cgiar.org/)". The observed gravity anomalies correlate extremely well with the main structural features of the basin. The fitted parabolic density model (PDM) to the measured density log data of the El Gredal-1 well, in the basin (I.T.G.E, 1990; SHELL, 1983; Gómez-Ortiz *et al.*, 2005), was used to build the EDM defined with $\Delta\rho_0$ = -0.573651 g/cm^3 and λ = 0.367852 km^{-1} (Ramamma et al., 2020). The two fitted density models (EDM and PDM) of the basin are shown in Fig. 5.6. The derived EDM was used to optimize the measured gravity data to deduce the basement configuration of the basin.

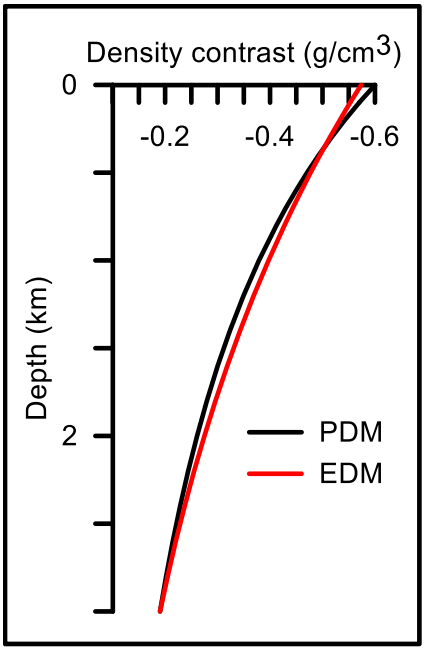


Fig. 5.6 Fitted density models (PDM, EDM) to the measured density log data from ElGredal-1 well, Almazán Basin (Ramamma et al., 2020). PDM was derived by Gómez-Ortiz et al. (2005).

For the present case gravity anomalies digitized into 25 X 18 nodes (Fig. 5.7a) were used in the inversion, keeping a tolerable misfit between the measured and theoretical anomalies at 0.05 mGal, for which the algorithm took 11 iterations. The initial misfit for the starting model was 0.681 mGal, which had attained to 0.098 mGal after the 10th iteration (Fig. 5.7c). Because there was an increase in the magnitude of data misfit subsequent to the 11th iteration (0.1027 mGal), the existing damping factor (9.76E-04) was doubled (1.95E-04) and the depth improvements were again estimated to update the model space and consequently the misfit. The new misfit corresponding to the improved model space had shown a magnitude higher than the misfit

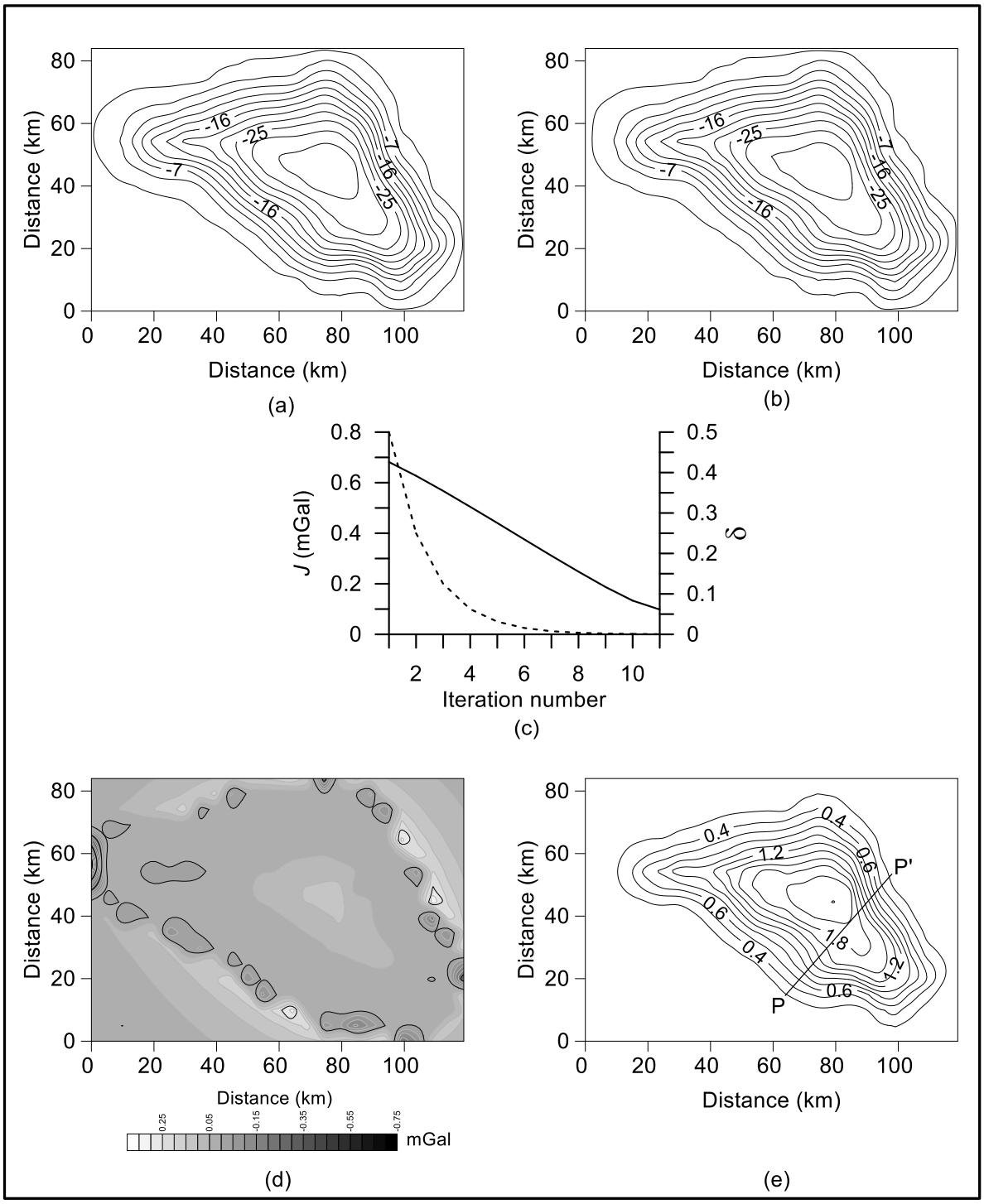


Fig. 5.7(a) Gravity anomalies of the Almazán Basin (Gómez-Ortiz et al., 2005), (b) theoretical gravity anomaly with EDM by present technique, (c) changes in misfit (solid line) and damping factor (dashed line) against iteration, (d) differences between the measured and theoretical gravity anomalies, (e) deduced depth structure of the basin. PP' is a profile along which seismic data interpretation had been reported (Bond, 1996).

obtained after the 11th iteration; hence, the exercise of increasing the damping, updating the model space was continued 14 times. At the end of the 14th turn the resulting damping factor had reached to a value of more than 12, thereby, the algorithm got terminated. The modeled gravity anomaly obtained after the 11th iteration (Fig. 5.7b) comply reasonably well with the observed anomaly (Fig. 5.7a). By and large, the left out differences between the measured and recalculated anomalies at the end of the concluding inversion were within ± 0.35 mGal over a large part of the basin (Fig. 5.7d).

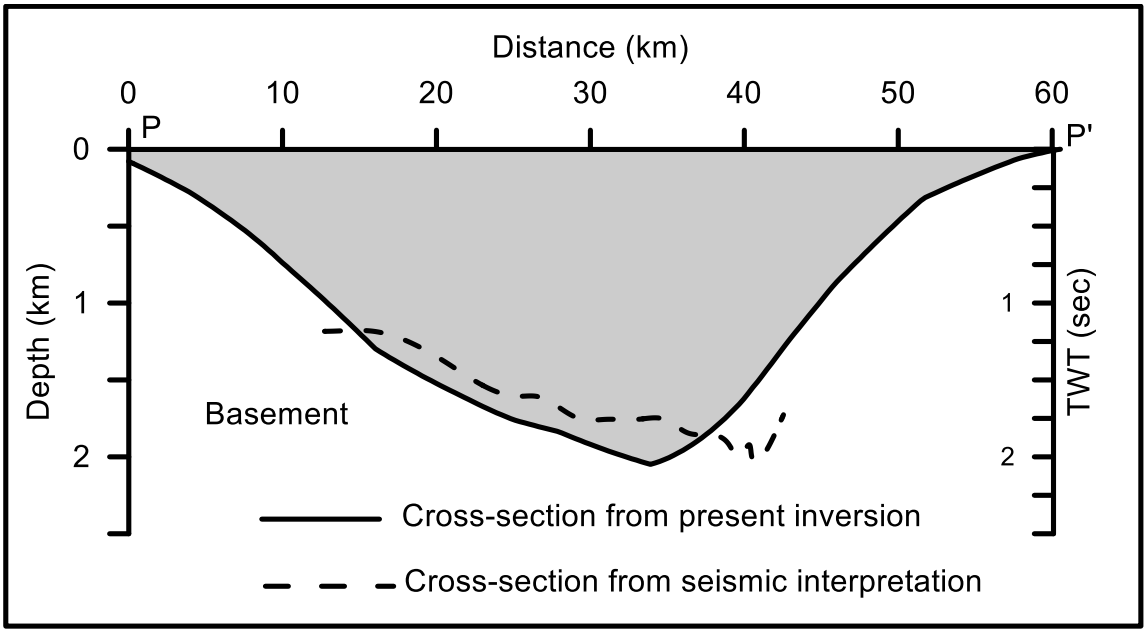


Fig. 5.8 Cross-section of the Almazán Basin obtained from Fig. 5.7e along the seismic profile, PP'. Note that seismic section is scaled as a function of two way travel time (after Bond, 1996).

The deduced configuration of the basin from measured gravity anomalies after the 11th iteration is shown in Fig. 5.7e. The inversion result has yielded a figure of 2.3 km for the maximum thickness of the basin at its depocentre (Fig. 5.7e). Interpretation of seismic data along a NE-SW profile across the basin (shown as PP' in Fig. 5.7e) has revealed 2.5 km for the maximum thickness of the basin (Bond, 1996). Fig. 5.8 shows the cross-section of the basin along the profile, PP' derived from Fig. 5.7e along which seismic interpretation was reported (Bond, 1996). The geometry of basement relief (Fig. 5.8) unveils that the basin is a typical half-graben structure with the master fault being located on the northeastern side. The persistence between the present interpretation and the one derived from seismic data has proved the practical applicability of the algorithm.

5.5 Results and discussion

i) The principles of optimization "(Chakravarthi *et al.*, 2017"; Ramamma et al., 2020) are used to design a new 3D inversion technique and related software to deduce configurations of density interfaces with EDM using spatially distributed gravity data set.

ii) The model space is described with a stack of vertical polygonal cross-sections, each one with unit thickness. Further, keeping in view the fact that EDM is more appropriate to explain the vertical variation of sediment density with depth (as evidenced from many field studies) the present technique uses this function in both initializing and improvising the structure.

iii) The proposed optimization technique is applied to analyse the anomalies produced by a model of typical intracratonic sedimentary basin in the presence of pseudorandom noise. Following this, to show the practical applicability, the same technique is also applied to interpret the real field gravity anomaly pertaining to the Almazán Basin in northeast Spain.

iv) In the case of synthetic anomalies with pseudorandom noise, the inversion technique had recovered the structure, however, with minor deviations that are within the tolerable limit.

v) In the case of field example, the estimated structure of the Almazán Basin from present technique is well correlated with the reported information derived from seismic data.

vi) The successful demonstration of the applicability of the technique both on synthetic noisy data and real world gravity anomalies proves its validity.

CHAPTER
SIX

Conclusions

This thesis is organized into six chapters. The research work presented in chapters – 2, 3, 4, and 5 reports the development of new interpretation techniques of sedimentary basin gravity anomalies making use of the exponential density model.

The highlight of these new techniques are primarily on the following points

1. Efficiency of 2.75D and 3D automatic modeling strategies and related software,
2. Efficacy of 2.75D and 3D inversion strategies and related software.

In the application of both automatic modeling and inversion techniques sedimentary basins are considered as inhomogeneous with the density within source volume varies exponentially with depth. New schemes are presented, in the spatial domain, to compute gravity responses of 2.75D and 3D sources with EDM. Accuracy of each such forward modeling is established against the corresponding analytic solution obtained over homogeneous sedimentary basin.

A total of four new interpretation techniques are developed and presented to interpret sedimentary basin gravity anomalies, among which two schemes make use of principles of automatic modeling and remaining two the principles of optimization. A total of four softwares, coded in Core Java, were also developed and presented. The advantage of these codes is that they can be installed and run on any platform, irrespective of machine's hardware and software. Displaying of interpreted results in animated versions during the process of analysis, in each case, is yet other notable feature of these softwares.

The applicability of each technique is illustrated on both theoretical models and real-world sedimentary basins.

Scope for future research

Interpretation of sedimentary basin gravity anomalies in the presence of interfering sources within the source volume is always challenging. It is anticipated that future research would likely to cater suitable algorithms to address this problem.

```
package com.polymodexp.view;

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

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

import com.polymodexp.control.POLYMODEXP_Controller;
import com.polymodexp.model.POLYMODEXP_CalculateValues;
import com.polymodexp.view.POLYMODEXP_MainPanel;
import com.polymodexp.view.POLYMODEXP_MainView;

public class POLYMODEXP_MainView extends Frame{


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

	public static void main(String s[])	{

		POLYMODEXP_MainView cm = new POLYMODEXP_MainView();
		cm.setSize(1280, 768);
		cm.addWindowListener(new WindowAdapter(){
			public void windowClosing(WindowEvent e){
				JFrame frame = null;

				int r =	JOptionPane.showConfirmDialog(
					frame,
					"Exit POLYMODEXP ?",
					"Confirm Exit ",
					JOptionPane.YES_NO_OPTION);
				if(r == JOptionPane.YES_OPTION ){
					if(POLYMODEXP_Controller.success==false){
					String fileName =
POLYMODEXP_CalculateValues.input_area_name+".jpg";
					File f = new File(fileName);
					f.delete();
					}
					System.exit(0);
				}
			}
		});
		cm.setTitle("POLYMODEXP");
		cm.setResizable(false);
		cm.add(new POLYMODEXP_MainPanel());
```

```
            cm.setVisible(true);

        }
}
```

```
package com.polymodexp.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;
import java.awt.TextField;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;

import javax.swing.JFileChooser;

import com.polymodexp.view.POLYMODEXP_MainPanel;
import com.polymodexp.view.POLYMODEXP_TableView;

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

public class POLYMODEXP_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[13];
        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 ;
/**Y-values*/
final static int Y_KM = 7;
/**STRIKE-values*/
final static int STRIKE_KM = 8;
/**ZMIN(km)*/
public static final int DEP_ZMIN = 9;
/**ZMAX(km)*/
public static final int DEP_ZMAX = 10;
/**Number of iteration values*/
public static final int NOB_ITER = 11;


public POLYMODEXP_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 2.5D 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 < 12; 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();
        POLYMODEXP_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("Offset (km)"));
        p_North.add(inputValues[7]);
        p_North.add(new Label("Half strike length (km)"));
        p_North.add(inputValues[8]);
        p_North.add(new Label("Minimum depth(km):"));
        p_North.add(inputValues[9]);
        p_North.add(new Label("Maximum depth(km):"));
        p_North.add(inputValues[10]);
        p_North.add(new Label("Iterations"));
        p_North.add(inputValues[11]);


        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.polymodexp.control.POLYMODEXP_Controller());
            actionButton[1].addActionListener(new
com.polymodexp.control.POLYMODEXP_Controller());
            actionButton[2].addActionListener(new
com.polymodexp.control.POLYMODEXP_Controller());
            actionButton[3].addActionListener(new
com.polymodexp.control.POLYMODEXP_Controller());
            actionButton[4].addActionListener(new
com.polymodexp.control.POLYMODEXP_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("Y_KM", inputValues[Y_KM].getText());
                  h_Map.put("STRIKE_KM", inputValues[STRIKE_KM].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];
                        //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());

POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.AREA_FE].setT
ext(cell.getContents());

                                        }

                                        if (type == CellType.NUMBER) {
                                                if (j == 1){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.PROFILE_NAM
E].setText(cell.getContents());

                                                }
                                                if (j == 2){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.N_OBS].setText
(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){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.SD_POLY].setT
ext(cell.getContents());

                    }
                    if (j == 6){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.LAMBDA_VAL].
setText(cell.getContents());

                    }
                    if (j == 7){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.STRIKE_KM].s
etText(cell.getContents());

                    }
                    if (j == 8){



POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.Y_KM].setText(
cell.getContents());
                    }
                    if (j == 9){


POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.DEP_ZMIN].set
Text(cell.getContents());

                    }
                    if (j == 10){
```

```
POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.DEP_ZMAX].se
tText(cell.getContents());
                                        }
                                        if (j == 11){

POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.NOB_ITER].set
Text(cell.getContents());

                                        }

                                        //System.out.println("I got a number "
                                        //  + cell.getContents());
                                    }

                                }
                            }

POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.X_KM].setText("
"+disval);

POLYMODEXP_MainPanel.inputValues[POLYMODEXP_MainPanel.NOB_GOB].set
Text(""+gobsval);
                        }
                    }
                    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[Y_KM].setText("");
        inputValues[STRIKE_KM].setText("");
        inputValues[NOB_GOB].setText("");
        inputValues[NOB_ITER].setText("");
        inputValues[AREA_FE].setText("");
        inputValues[PROFILE_NAME].setText("");
```

```
	}

}
```

```
package com.polymodexp.view;

import java.awt.Dimension;

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

public class POLYMODEXP_TableView {

	public static  void populateEastPanel(Object rowData[][]) {
		com.polymodexp.view.POLYMODEXP_MainPanel.p_East.removeAll();
		Object columnNames[] = {"Distance(km)", "Observed anomalies (mGal)",
"Calculated anomalies (mGal)","Depth(km)"};
		JTable table = new JTable(rowData, columnNames);
		table.setPreferredScrollableViewportSize(new Dimension(300,550));

		JScrollPane scrollPane = new JScrollPane(table);
		scrollPane.setAutoscrolls(true);

com.polymodexp.view.POLYMODEXP_MainPanel.p_East.add(scrollPane);
		com.polymodexp.view.POLYMODEXP_MainPanel.p_East.validate();

com.polymodexp.view.POLYMODEXP_MainPanel.p_East.setVisible(true);

	}

}
```

```
package com.polymodexp.view;

import java.applet.Applet;
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.polymodexp.model.POLYMODEXP_CalculateValues;
import com.polymodexp.util.POLYMODEXP_Utility;

public class POLYMODEXP_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)
POLYMODEXP_Utility.findMaximumNumber1(POLYMODEXP_CalculateValues.inpu
t_x_km); //i_no_obs;//
		maxY =
POLYMODEXP_Utility.findMaximumNumber(POLYMODEXP_CalculateValues.input
_nob_gob);
		inidep  =
POLYMODEXP_Utility.findMaximumNumber1(POLYMODEXP_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("l", 600, 308);

g.drawString(""+df.format(POLYMODEXP_CalculateValues.input_x_km[POLYMODE
XP_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)
(POLYMODEXP_CalculateValues.input_x_km[POLYMODEXP_CalculateValues.inpu
t_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("l",(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 <= POLYMODEXP_CalculateValues.input_n_obs; k++){

                  xpoint = (float)(450 *
POLYMODEXP_CalculateValues.input_x_km[k]/ maxX) ;
                  ypoint = (float)(250 * POLYMODEXP_CalculateValues.gc[k]/ maxY);
                  gypoint = (float)(250 *
POLYMODEXP_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 <= POLYMODEXP_CalculateValues.input_n_obs; k++){

                  xpoint = (float)(450 *
POLYMODEXP_CalculateValues.input_x_km[k]/ maxX);
                  zpoint = (float)(250 * POLYMODEXP_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 =
POLYMODEXP_Utility.findMaximumNumber1(POLYMODEXP_CalculateValues.o_fu
nct);
            int ini =
POLYMODEXP_Utility.findMaximumNumber(POLYMODEXP_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 <= POLYMODEXP_CalculateValues.o_iter; i++) {

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

                  if(i==POLYMODEXP_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(POLYMODEXP_CalculateValues.o_funct[1]),
780, 70);
            g2.drawString("
"+d1.format(POLYMODEXP_CalculateValues.o_funct[POLYMODEXP_CalculateValu
es.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 =
POLYMODEXP_Utility.findMaximumNumber1(POLYMODEXP_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(POLYMODEXP_CalculateValues.vsd[1] )) / maxOb1 ) );
            float prevy = 300;
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= POLYMODEXP_CalculateValues.count; i++) {

                  xpoint = (float)( 90 *
Math.abs(POLYMODEXP_CalculateValues.vsd[i]) / maxOb1 );
                  ypoint = (float)( 250 * POLYMODEXP_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(POLYMODEXP_CalculateValues.vsd[1] ),805+
(float) ( ( 90 * ( Math.abs(POLYMODEXP_CalculateValues.vsd[1] )) / maxOb1 ) ) ,
300 );

g2.drawString(""+d1.format(POLYMODEXP_CalculateValues.vsd[POLYMODEXP_C
alculateValues.count] ),820+ (float) ( ( 90 *
( Math.abs(POLYMODEXP_CalculateValues.vsd[POLYMODEXP_CalculateValues.c
ount] )) / 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.polymodexp.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.polymodexp.model.POLYMODEXP_CalculateValues;
import com.polymodexp.util.POLYMODEXP_Utility;
import com.polymodexp.view.POLYMODEXP_DrawGraph;
import com.polymodexp.view.POLYMODEXP_MainPanel;

public class POLYMODEXP_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 input_strike_km;
	public static double input_y_km;
	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 =
POLYMODEXP_Utility.convertInteger((String)h_Map.get("N_OBS"));
			input_sd_poly =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("SD_POLY"));
			input_lambda_val =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("LAMBDA_VAL"));
			input_zmin_km =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("DEP_ZMIN"));
			input_zmax_km =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("DEP_ZMAX"));
			input_x_km =
POLYMODEXP_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
			input_nob_gob =
POLYMODEXP_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
			input_nob_iter =
POLYMODEXP_Utility.convertInteger((String)h_Map.get("NOB_ITER"));
			input_strike_km =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("STRIKE_KM"));
			input_y_km =
POLYMODEXP_Utility.convertDouble((String)h_Map.get("Y_KM"));
			input_area_name =
POLYMODEXP_Utility.convertString((String)h_Map.get("AREA_FE"));
			input_profile =
POLYMODEXP_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)));

            }

            gBasin(input_n_obs, input_x_km, z, input_sd_poly, input_lambda_val, gk,
input_strike_km, input_y_km, 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);
            //System.out.println(funct1);
            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;

                }
                gBasin(input_n_obs, input_x_km, z, input_sd_poly,
input_lambda_val, gk, input_strike_km, input_y_km, 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=POLYMODEXP_CalculateValues.input_zmin_km;
            double z2 =
POLYMODEXP_Utility .findMaximumNumber1(POLYMODEXP_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 [] gBasin(int n, double []x, double []zv, double sd, double la,
double gk, double hafstr, double offset, 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];
        double []effy = new double[3];
        double []gg2 = new double[3];
        for(int JJ = 1 ; JJ <= n ;JJ++){
            gc[JJ] = 0.0;
        }

        effy[1] = hafstr - offset;
        effy[2] = hafstr + offset;

        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]));
                        for (int kk = 1; kk <= 2; kk++) {
                            double y2 = effy[kk];
                            double a = xx[i] - zv[i] * ct;
                            double anum = y2 *(a + zt[jz] * ct);
                            double den1 = Math.sqrt(Math.pow(a + zt[jz] * ct, 2)
+ Math.pow(y2, 2) + Math.pow(zt[jz], 2));
                            double den = zt[jz] * den1;
                            gg2[kk]= -13.3333 * dc * Math.atan(anum / den);
                        }
                        gs[jz]= (gg2[1] + gg2[2]) / 2;

                    }
                    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;
		double sum2 = 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,double []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 POLYMODEXP_DrawGraph dg = new
POLYMODEXP_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( POLYMODEXP_CalculateValues.input_area_name+".jpg");
                    ImageIO.write(buffer, "jpg", os);
                    os.close();

                    String path =
POLYMODEXP_CalculateValues.input_area_name+".jpg";
                    image = ImageIO.read(new File(path));

                    Graphics g_image = POLYMODEXP_MainPanel.img.getGraphics();
                    g_image.drawImage(image, -80, -20,image.getWidth(),
image.getHeight(),dg);

                    MouseListener ml3 = new MouseAdapter(){
                          public void mouseClicked(MouseEvent e){
                                Graphics g_image =
POLYMODEXP_MainPanel.img.getGraphics();
                                g_image.drawImage(image, -80,-40,image.getWidth(),
image.getHeight(),dg);
                          }
                    };
                    POLYMODEXP_MainPanel.img.addMouseListener(ml3);
              }
              catch (Exception e2) {

                    e2.printStackTrace();
              }
        }
}
```

```
package com.polymodexp.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.polymodexp.control.POLYMODEXP_PrintValues;
import com.polymodexp.model.POLYMODEXP_CalculateValues;
import com.polymodexp.view.POLYMODEXP_DrawGraph;
import com.polymodexp.view.POLYMODEXP_MainPanel;
import com.polymodexp.view.POLYMODEXP_TableView;

public class POLYMODEXP_Controller implements ActionListener{

	POLYMODEXP_DrawGraph dg = new POLYMODEXP_DrawGraph();
	Object rowdata[][]={};
	public static boolean success=false;
	public void actionPerformed(ActionEvent ae) {

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

			com.polymodexp.model.POLYMODEXP_CalculateValues cv
= new com.polymodexp.model.POLYMODEXP_CalculateValues();

cv.getAnamolyValues(com.polymodexp.view.POLYMODEXP_MainPanel.capture
Values());

com.polymodexp.view.POLYMODEXP_TableView.populateEastPanel(POLYMODEX
P_CalculateValues.obj);

com.polymodexp.view.POLYMODEXP_MainPanel.p_East.repaint();
		}else if(ae.getActionCommand().equals("Save and Print")){
			try
			{
				POLYMODEXP_PrintValues.printGraphValues();
			}
			catch(Exception e1) {
				e1.printStackTrace();
			}
		}else if(ae.getActionCommand().equals("Load data")){
			try {
				POLYMODEXP_MainPanel.loadData1();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
```

```
                }
            }
            else if(ae.getActionCommand().equals("Clear")){
                POLYMODEXP_MainPanel.clearDefaultValues();


POLYMODEXP_MainPanel.clearPanel(POLYMODEXP_MainPanel.img);
                POLYMODEXP_TableView.populateEastPanel(rowdata);
            }
            else if(ae.getActionCommand().equals("Exit")){
                JFrame frame = null;

                int r =    JOptionPane.showConfirmDialog(
                           frame,
                           "Exit POLYMODEXP ?",
                           "Confirm Exit ",
                           JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                     if(success==false){
                           String fileName =
POLYMODEXP_CalculateValues.input_area_name+".jpg";
                           File f = new File(fileName);
                           f.delete();
                     }
                     System.exit(0);
                }

            }

      }

}
```

```
package com.polymodexp.control;

import java.io.File;
import java.io.FileWriter;
import java.text.DecimalFormat;

import javax.swing.JFileChooser;

import com.polymodexp.control.POLYMODEXP_Controller;
```

```
import com.polymodexp.model.POLYMODEXP_CalculateValues;

public class POLYMODEXP_PrintValues {
      public static void printGraphValues() throws Exception {

            try{
                  String current = System.getProperty("user.dir");
                  File img_file = new
File(POLYMODEXP_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());
                              POLYMODEXP_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 = '"+ POLYMODEXP_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:-
"+POLYMODEXP_CalculateValues.input_area_name+"</th> </tr>");
                              DecimalFormat df =new
DecimalFormat("0.###");
                              myWriter.write(" <tr><th colspan = 4>
PROFILE NUMBER:-"+"     "+POLYMODEXP_CalculateValues.input_profile+"
</th></tr>");
                              myWriter.write(" <tr><th colspan = 4>
ITERATION:-"+"     "+POLYMODEXP_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 <=
POLYMODEXP_CalculateValues.input_n_obs; K++){
```

```
                                    myWriter.write("<tr> <td>" +
POLYMODEXP_CalculateValues.input_x_km[K]+"</td>
<td>"+df.format(POLYMODEXP_CalculateValues.input_nob_gob[K])+"</td>
<td>"+df.format(POLYMODEXP_CalculateValues.gc[K])+"</td>
<td>"+df.format(POLYMODEXP_CalculateValues.z[K])+"</td></tr>");
                              }

                                    myWriter.close();
                              }
                        }
                        else
                        {
                              //pops up error message
                        }
                  }
                  catch(Exception e1) {

                        e1.printStackTrace();
                  }
            }
}
```

```
package com.polymodexp.util;

import com.polymodexp.model.POLYMODEXP_CalculateValues;

public class POLYMODEXP_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 = POLYMODEXP_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;
      }
}
```

```
package com.polyinvexp.view;

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

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

import com.polyinvexp.control.POLYINVEXP_Controller;
import com.polyinvexp.model.POLYINVEXP_CalculateValues;
import com.polyinvexp.view.POLYINVEXP_MainPanel;
import com.polyinvexp.view.POLYINVEXP_MainView;

public class POLYINVEXP_MainView extends Frame{


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

	public static void main(String s[])	{

		POLYINVEXP_MainView cm = new POLYINVEXP_MainView();
		cm.setSize(1280, 768);
		cm.addWindowListener(new WindowAdapter(){
			public void windowClosing(WindowEvent e){
				JFrame frame = null;

				int r =	JOptionPane.showConfirmDialog(
					frame,
					"Exit POLYINVEXP ?",
					"Confirm Exit ",
					JOptionPane.YES_NO_OPTION);
				if(r == JOptionPane.YES_OPTION ){
					if(POLYINVEXP_Controller.success==false){
					String fileName =
POLYINVEXP_CalculateValues.input_area_name+".jpg";
					File f = new File(fileName);
					f.delete();
					}
					System.exit(0);
				}
			}
		});
		cm.setTitle("POLYINVEXP");
		cm.setResizable(false);
		cm.add(new POLYINVEXP_MainPanel());
```

```
            cm.setVisible(true);

        }
}
```

```
package com.polyinvexp.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;
import java.awt.TextField;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;

import javax.swing.JFileChooser;

import com.polyinvexp.view.POLYINVEXP_MainPanel;
import com.polyinvexp.view.POLYINVEXP_TableView;

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

public class POLYINVEXP_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[13];
        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 ;
        /**Y-values*/
        final static int Y_KM = 7;
        /**STRIKE-values*/
        final static int STRIKE_KM = 8;
        /**ZMIN(km)*/
        public static final int DEP_ZMIN = 9;
        /**ZMAX(km)*/
        public static final int DEP_ZMAX = 10;
        /**Number of iteration values*/
        public static final int NOB_ITER = 11;


        public POLYINVEXP_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 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 < 12; 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();
        POLYINVEXP_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("Offset (km)"));
        p_North.add(inputValues[7]);
        p_North.add(new Label("Half strike length (km)"));
        p_North.add(inputValues[8]);
        p_North.add(new Label("Minimum depth(km):"));
        p_North.add(inputValues[9]);
        p_North.add(new Label("Maximum depth(km):"));
        p_North.add(inputValues[10]);
        p_North.add(new Label("Iterations"));
        p_North.add(inputValues[11]);
        //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.polyinvexp.control.POLYINVEXP_Controller());
            actionButton[1].addActionListener(new
com.polyinvexp.control.POLYINVEXP_Controller());
            actionButton[2].addActionListener(new
com.polyinvexp.control.POLYINVEXP_Controller());
            actionButton[3].addActionListener(new
com.polyinvexp.control.POLYINVEXP_Controller());
            actionButton[4].addActionListener(new
com.polyinvexp.control.POLYINVEXP_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("Y_KM", inputValues[Y_KM].getText());
                  h_Map.put("STRIKE_KM", inputValues[STRIKE_KM].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];
                //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());

POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.AREA_FE].setText
(cell.getContents());

                        }

                        if (type == CellType.NUMBER) {
                            if (j == 1){


POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.PROFILE_NAME].
setText(cell.getContents());

                            }
                            if (j == 2){
```

```
POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.N_OBS].setText(ce
ll.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){


POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.SD_POLY].setText
(cell.getContents());

                                        }
                                        if (j == 6){


POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.LAMBDA_VAL].set
Text(cell.getContents());

                                        }
                                        if (j == 7){


POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.STRIKE_KM].setT
ext(cell.getContents());

                                        }
                                        if (j == 8){



POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.Y_KM].setText(cell
.getContents());
                                        }
```

```
                                        if (j == 9){

POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.DEP_ZMIN].setTex
t(cell.getContents());

                                        }
                                        if (j == 10){


POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.DEP_ZMAX].setTe
xt(cell.getContents());
                                        }
                                        if (j == 11){

POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.NOB_ITER].setTex
t(cell.getContents());

                                        }

                                        //System.out.println("I got a number "
                                        //  + cell.getContents());
                                }

                        }
                }

POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.X_KM].setText(""+
disval);

POLYINVEXP_MainPanel.inputValues[POLYINVEXP_MainPanel.NOB_GOB].setTex
t(""+gobsval);
                }
        }
        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[Y_KM].setText("");
            inputValues[STRIKE_KM].setText("");
            inputValues[NOB_GOB].setText("");
            inputValues[NOB_ITER].setText("");
            inputValues[AREA_FE].setText("");
            inputValues[PROFILE_NAME].setText("");
      }


}
```

```
package com.polyinvexp.view;

import java.awt.Dimension;

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

public class POLYINVEXP_TableView {
      public static  void populateEastPanel(Object rowData[][]) {
            com.polyinvexp.view.POLYINVEXP_MainPanel.p_East.removeAll();
            Object columnNames[] = {"Distance(km)", "Observed anomalies (mGal)",
"Calculated anomalies (mGal)","Depth(km)"};
            JTable table = new JTable(rowData, columnNames);
            table.setPreferredScrollableViewportSize(new Dimension(300,550));

            JScrollPane scrollPane = new JScrollPane(table);
            scrollPane.setAutoscrolls(true);
            com.polyinvexp.view.POLYINVEXP_MainPanel.p_East.add(scrollPane);
            com.polyinvexp.view.POLYINVEXP_MainPanel.p_East.validate();
            com.polyinvexp.view.POLYINVEXP_MainPanel.p_East.setVisible(true);

      }

}
```

```
package com.polyinvexp.view;
```

```
import java.applet.Applet;
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.polyinvexp.model.POLYINVEXP_CalculateValues;
import com.polyinvexp.util.POLYINVEXP_Utility;

public class POLYINVEXP_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)
POLYINVEXP_Utility.findMaximumNumber1(POLYINVEXP_CalculateValues.input_x
_km); //i_no_obs;//
		maxY =
POLYINVEXP_Utility.findMaximumNumber(POLYINVEXP_CalculateValues.input_no
b_gob);
```

```
            inidep  =
POLYINVEXP_Utility.findMaximumNumber1(POLYINVEXP_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("I", 600, 308);

g.drawString(""+df.format(POLYINVEXP_CalculateValues.input_x_km[POLYINVEXP
_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)
(POLYINVEXP_CalculateValues.input_x_km[POLYINVEXP_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("I",(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 <= POLYINVEXP_CalculateValues.input_n_obs; k++){

			xpoint = (float)(450 * POLYINVEXP_CalculateValues.input_x_km[k]/
maxX) ;
			ypoint = (float)(250 * POLYINVEXP_CalculateValues.gc[k]/ maxY);
			gypoint = (float)(250 *
POLYINVEXP_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 <= POLYINVEXP_CalculateValues.input_n_obs; k++){

			xpoint = (float)(450 * POLYINVEXP_CalculateValues.input_x_km[k]/
maxX);
			zpoint = (float)(250 * POLYINVEXP_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 =
POLYINVEXP_Utility.findMaximumNumber1(POLYINVEXP_CalculateValues.o_funct
);
            int ini =
POLYINVEXP_Utility.findMaximumNumber(POLYINVEXP_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 <= POLYINVEXP_CalculateValues.o_iter; i++) {

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

                if(i==POLYINVEXP_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(POLYINVEXP_CalculateValues.o_funct[1]),
780, 70);
            g2.drawString("
"+d1.format(POLYINVEXP_CalculateValues.o_funct[POLYINVEXP_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 =
POLYINVEXP_Utility.findMaximumNumber1(POLYINVEXP_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(POLYINVEXP_CalculateValues.vsd[1] )) / maxOb1 ) );
            float prevy = 300;
            float xpoint = 0;
            float ypoint = 0;

            for (int i = 1; i <= POLYINVEXP_CalculateValues.count; i++) {

                  xpoint = (float)( 90 *
Math.abs(POLYINVEXP_CalculateValues.vsd[i]) / maxOb1 );
                  ypoint = (float)( 250 * POLYINVEXP_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(POLYINVEXP_CalculateValues.vsd[1] ),805+
(float) ( ( 90 * ( Math.abs(POLYINVEXP_CalculateValues.vsd[1] )) / maxOb1 ) ) ,
300 );

g2.drawString(""+d1.format(POLYINVEXP_CalculateValues.vsd[POLYINVEXP_Calc
ulateValues.count] ),820+ (float) ( ( 90 *
( Math.abs(POLYINVEXP_CalculateValues.vsd[POLYINVEXP_CalculateValues.coun
t] )) / 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.polyinvexp.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.polyinvexp.model.POLYINVEXP_CalculateValues;
import com.polyinvexp.util.POLYINVEXP_Utility;
```

```
import com.polyinvexp.view.POLYINVEXP_DrawGraph;
import com.polyinvexp.view.POLYINVEXP_MainPanel;

public class POLYINVEXP_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 input_strike_km;
	public static double input_y_km;
	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 =
POLYINVEXP_Utility.convertInteger((String)h_Map.get("N_OBS"));
			input_sd_poly =
POLYINVEXP_Utility.convertDouble((String)h_Map.get("SD_POLY"));
			input_lambda_val =
POLYINVEXP_Utility.convertDouble((String)h_Map.get("LAMBDA_VAL"));
			input_zmin_km =
POLYINVEXP_Utility.convertDouble((String)h_Map.get("DEP_ZMIN"));
			input_zmax_km =
POLYINVEXP_Utility.convertDouble((String)h_Map.get("DEP_ZMAX"));
			input_x_km =
POLYINVEXP_Utility.convertDoubleArray((String)h_Map.get("X_KM"));
			input_nob_gob =
POLYINVEXP_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
			input_strike_km =
```

```
POLYINVEXP_Utility.convertDouble((String)h_Map.get("STRIKE_KM"));
                    input_y_km =
POLYINVEXP_Utility.convertDouble((String)h_Map.get("Y_KM"));
                    input_nob_iter =
POLYINVEXP_Utility.convertInteger((String)h_Map.get("NOB_ITER"));
                    input_area_name =
POLYINVEXP_Utility.convertString((String)h_Map.get("AREA_FE"));
                    input_profile =
POLYINVEXP_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,input_strike_
km,input_y_km,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);
        System.out.println(funct1);
        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,input_strike
_km,input_y_km,g1);
                par1[i] = par[i]-dpar/2;

GBSN(input_n_obs,input_x_km,par1,input_sd_poly,input_lambda_val,gk,input_strike
_km,input_y_km,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;


                    }

GBSN(input_n_obs,input_x_km,par2,input_sd_poly,input_lambda_val,gk,input_strike
_km,input_y_km,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 =
POLYINVEXP_Utility .findMaximumNumber1(POLYINVEXP_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(-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 hafstr,double offset,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];
            double []effy = new double[3];
            double []gg2 = new double[3];

            zv[1] = 0.0001;
            zv[n] = 0.0001;
            effy[1] = hafstr - offset;
            effy[2] = hafstr + offset;
            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 ll 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]));
                        for (int kk = 1; kk <= 2; kk++) {
                            double y2 = effy[kk];
                            double a = xx[i] - zv[i] * ct;
                            double anum = y2 *(a + zt[jz] * ct);
                            double den1 = Math.sqrt(Math.pow(a + zt[jz] * ct, 2)
+ Math.pow(y2, 2) + Math.pow(zt[jz], 2));
                            double den = zt[jz] * den1;
                            gg2[kk]= -13.3333*dc*Math.atan(anum / den);
                        }
                        gs[jz]= (gg2[1] + gg2[2]) / 2;
                    }
                    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;
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[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 POLYINVEXP_DrawGraph dg = new POLYINVEXP_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( POLYINVEXP_CalculateValues.input_area_name+".jpg");
                  ImageIO.write(buffer, "jpg", os);
                  os.close();

                  String path =
POLYINVEXP_CalculateValues.input_area_name+".jpg";
                  image = ImageIO.read(new File(path));

                  Graphics g_image = POLYINVEXP_MainPanel.img.getGraphics();
                  g_image.drawImage(image, -80, -20, image.getWidth(),
image.getHeight(), dg);
                  MouseListener ml3 = new MouseAdapter(){
                        public void mouseClicked(MouseEvent e){
                              Graphics g_image =
POLYINVEXP_MainPanel.img.getGraphics();
                              g_image.drawImage(image, -80,-40,image.getWidth(),
image.getHeight(),dg);
                        }
                  };
                  POLYINVEXP_MainPanel.img.addMouseListener(ml3);
```

```
            }
            catch (Exception e2) {

                  e2.printStackTrace();
            }
      }

}
```

```
package com.polyinvexp.control;

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.polyinvexp.model.POLYINVEXP_CalculateValues;
import com.polyinvexp.view.POLYINVEXP_DrawGraph;
import com.polyinvexp.view.POLYINVEXP_MainPanel;

public class POLYINVEXP_Controller implements ActionListener {
      POLYINVEXP_DrawGraph dg = new POLYINVEXP_DrawGraph();
      public static boolean success=false;
      Object rowdata[][]={};
      public void actionPerformed(ActionEvent ae) {

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

                  com.polyinvexp.model.POLYINVEXP_CalculateValues cv = new
com.polyinvexp.model.POLYINVEXP_CalculateValues();

cv.getAnamolyValues(com.polyinvexp.view.POLYINVEXP_MainPanel.captureValues
());

com.polyinvexp.view.POLYINVEXP_TableView.populateEastPanel(POLYINVEXP_C
alculateValues.obj);
                  com.polyinvexp.view.POLYINVEXP_MainPanel.p_East.repaint();
```

```
        }else if(ae.getActionCommand().equals("Save and Print")){
            try {
                String current = System.getProperty("user.dir");
                File img_file = new
File(POLYINVEXP_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 = '"+
POLYINVEXP_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:- "+POLYINVEXP_CalculateValues.input_area_name+"</th> </tr>");
                        DecimalFormat df =new DecimalFormat("0.###");
                        myWriter.write(" <tr><th colspan = 4> PROFILE
NUMBER:-"+"     "+POLYINVEXP_CalculateValues.input_profile+" </th></tr>");
                        myWriter.write(" <tr><th colspan = 4>
ITERATION:-"+"     "+POLYINVEXP_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 <=
POLYINVEXP_CalculateValues.input_n_obs; K++){

                            myWriter.write("<tr> <td>" +
POLYINVEXP_CalculateValues.input_x_km[K]+"</td>
<td>"+df.format(POLYINVEXP_CalculateValues.input_nob_gob[K])+"</td>
<td>"+df.format(POLYINVEXP_CalculateValues.gc[K])+"</td>
<td>"+df.format(POLYINVEXP_CalculateValues.z[K])+"</td><td>"+"</td></tr>");
                        }

                    }
                    myWriter.close();
                }
            }
            catch (Exception e) {

                e.printStackTrace();
```

```
                }

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


POLYINVEXP_MainPanel.clearPanel(POLYINVEXP_MainPanel.img);

com.polyinvexp.view.POLYINVEXP_TableView.populateEastPanel(rowdata);
            }

            else if(ae.getActionCommand().equals("Exit")){
                JFrame frame = null;

                int r =     JOptionPane.showConfirmDialog(
                            frame,
                            "Exit POLYINVEXP ?",
                            "Confirm Exit ",
                            JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                    if(success==false){
                        String fileName =
POLYINVEXP_CalculateValues.input_area_name+".jpg";
                        File f = new File(fileName);
                        f.delete();
                    }
                    System.exit(0);
                }
            }

      }


}
```

```
package com.polyinvexp.util;

import com.polyinvexp.model.POLYINVEXP_CalculateValues;

public class POLYINVEXP_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 = POLYINVEXP_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;
      }
}
```

```
package com.polymod3d.view;

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

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

import com.polymod3d.control.POLYMOD3D_Controller;
import com.polymod3d.model.POLYMOD3D_CalculateValues;

public class POLYMOD3D_MainView  extends Frame {

 /**
          AUTOMATIC MODELING 3D POLYGON
  *
  */
         private static final long serialVersionUID = 1L;

 public static void main(String s[])
 {
                  POLYMOD3D_MainView cm = new POLYMOD3D_MainView();
           cm.setSize(1280, 768);
           cm.addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                             JFrame frame = null;

                             int r =           JOptionPane.showConfirmDialog(
                                       frame,
                                                "Exit POLYMOD3D ?",
                                       "Confirm Exit ",
                                       JOptionPane.YES_NO_OPTION);
                             if(r == JOptionPane.YES_OPTION ){
                                       if(POLYMOD3D_Controller.success==false){
                                       String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Observed anomaly.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                }
                                       if(POLYMOD3D_Controller.success1==false){
                                       String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Modeled anomaly.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                }
                                       if(POLYMOD3D_Controller.success2==false){
                                       String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Estimated Depth.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                }
                                       if(POLYMOD3D_Controller.success3==false){
                                       String fileName = "Index.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                }
                                       System.exit(0);
                             }
```

```
                }
        });
                cm.setTitle("POLYMOD3D");
        cm.setResizable(false);
                cm.add(new POLYMOD3D_MainPanel());
                cm.setVisible(true);

 }

}
```

```
package com.polymod3d.view;

import java.awt.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.HashMap;

import javax.swing.JFileChooser;

public class POLYMOD3D_MainPanel extends Panel {

 /**
  *
  */
        private static final long serialVersionUID = 1L;
 Panel p_North;
        static Panel p_West;
 public static Panel p_East;
 static Panel p_South;
 public static Panel p_Center;

        public static TextArea img = new TextArea(10,50);
        public static TextArea img1 = new TextArea(10,50);
        public static TextArea img2 = new TextArea(10,50);
        static TextField inputValues [] = new TextField[13];

 Button actionButton[] = new Button[7];
        Object rowdata[][]={};

 final static int NUMBER_COMPONENTS          = 6;
 final static int MIN_X_STEPS =   2,
                  MIN_Y_STEPS =   2,
                  MAX_X_STEPS = 100,
                  MAX_Y_STEPS = 100,
                  N_CONTOURS=10;
 static String              contourValuesTitle,infoStrX,infoStrY,
 errParse,errLog,errComp,errEqual,
        errExpect,errEOF,errBounds;

 public static ContourPlot thePlot              =      new
ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);
 public static ContourPlot thePlot1             =    new
ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);
```

```
static ContourPlot thePlot2          =    new ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);

final static int AREA_FE = 0;
        final static int MAT_NY = 1;
        final static int MAT_NX = 2;
final static int DIS_DY = 3;
final static int DIS_DX = 4;
        final static int SD_POLY = 5;
final static int ALPHA_ST = 6 ;
        final static int OFF_ZLT = 7;
final static int NOB_GOB = 8;
final static int N_OBS = 9 ;
final static int NUM_CON = 10;
final static int NUM_CON1 = 11;

//final static int PROFILE_NAME = 11;

        public POLYMOD3D_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();

         for(int i = 0; i < 13; i++){
                        inputValues[i] = new TextField();
         }
         actionButton[0] = new Button("Modeling");
         actionButton[1] = new Button("Contour");
         actionButton[2] = new Button("Save & Print");
         actionButton[3] = new Button("Load data");
         actionButton[4] = new Button("Clear");
         actionButton[5] = new Button("Exit");

         this.populateNorthPanel();

                POLYMOD3D_TableView.populateEastPanel(rowdata);
                this.add(p_North, BorderLayout.NORTH);
                p_Center.setSize(1000, 760);
                this.add(p_Center, BorderLayout.CENTER);
                //this.populateCentre();
                this.add(p_East, BorderLayout.EAST);
                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("No. of profiles along Y axis"));
                p_North.add(inputValues[1]);
         p_North.add(new Label("No. of observations along x axis"));
                p_North.add(inputValues[2]);
                p_North.add(new Label("Profile spacing along Y axis(km)"));
                p_North.add(inputValues[3]);
```

```
        p_North.add(new Label("Station spacing along X axis(km)"));
                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 (gm/cc/km)"));
                p_North.add(inputValues[6]);



        p_North.add(new Label("Maximum limiting depth(km)"));
                p_North.add(inputValues[7]);




        p_North.add(new Label("Observed anomalies (mGal)"));
                p_North.add(inputValues[8]);

        p_North.add(new Label("Number of iterations"));
                p_North.add(inputValues[9]);


        p_North.add(new Label("Number of contours for gravity anomalies"));
                p_North.add(inputValues[10]);

        p_North.add(new Label("Number of depth contours "));
                p_North.add(inputValues[11]);

        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.polymod3d.control.POLYMOD3D_Controller());
        actionButton[1].addActionListener(new
com.polymod3d.control.POLYMOD3D_Controller());
        actionButton[2].addActionListener(new
com.polymod3d.control.POLYMOD3D_Controller());
        actionButton[3].addActionListener(new
com.polymod3d.control.POLYMOD3D_Controller());
        actionButton[4].addActionListener(new
com.polymod3d.control.POLYMOD3D_Controller());
        actionButton[5].addActionListener(new
com.polymod3d.control.POLYMOD3D_Controller());

 }

       public static HashMap captureValues(){

        HashMap h_Map = new HashMap();
        try {

                        h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
                        h_Map.put("MAT_NY", inputValues[MAT_NY].getText());
                        h_Map.put("MAT_NX", inputValues[MAT_NX].getText());
                        h_Map.put("DIS_DY", inputValues[DIS_DY].getText());
```

```
                        h_Map.put("DIS_DX", inputValues[DIS_DX].getText());
                        h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
                        h_Map.put("ALPHA_ST",inputValues[ALPHA_ST].getText());
                        h_Map.put("OFF_ZLT", inputValues[OFF_ZLT].getText());
                        h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
                        h_Map.put("N_OBS", inputValues[N_OBS].getText());
                        h_Map.put("NUM_CON", inputValues[NUM_CON].getText());
                        h_Map.put("NUM_CON1", inputValues[NUM_CON1].getText());
                        //h_Map.put("PROFILE_NAME",inputValues[PROFILE_NAME].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, 35, 1280, 650);
}



public static void loadData(){
        try{

                        String current = System.getProperty("user.dir");
                        JFileChooser chooser=new  JFileChooser(current);
                        int returnVal = chooser.showOpenDialog(null);
                 int count = 0;
                        if(returnVal == JFileChooser.APPROVE_OPTION) {
                                File f = chooser.getSelectedFile();
                         Buff          eredReader br=new BufferedReader(new FileReader(f));
                         String st;
                                st = br.readLine();
                         count++;
                         while((st) != null){

                                  try {

                                          if (count == 1){


POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.AREA_FE].setText(""+st);

                                          }

                                          if (count == 2){

POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.MAT_NY].setText(""+st);


                                          }

                                          if (count == 3){
```

```
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.MAT_NX].setText(""+st);
                                        }
                                        if (count == 4){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.DIS_DY].setText(""+st);
                                        }
                                        if (count == 5){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.DIS_DX].setText(""+st);
                                        }


                                        if (count == 6){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.SD_POLY].setText(""+st);
                                        }
                                        if (count == 7){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.ALPHA_ST].setText(""+st);
                                        }
                                        if (count == 8){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.OFF_ZLT].setText(""+st);
                                        }


                                        if (count == 9){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.NOB_GOB].setText(""+st);
                                        }

                                        if (count == 10){
POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.N_OBS].setText(""+st);
                                        }
```

```
                                        if (count == 11){

POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.NUM_CON].setText(""+st);

                                        }

                                        if (count == 12){

POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.NUM_CON1].setText(""+st);

                                        }

                                        /*          if (count == 12){

POLYMOD3D_MainPanel.inputValues[POLYMOD3D_MainPanel.PROFILE_NAME].setText(""+st);

                                        }
                                         */

                                }
                                catch(Exception e) {

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

        }
}


        public static void clearDefaultValues(){

                inputValues[N_OBS].setText("");
                inputValues[SD_POLY].setText("");
                inputValues[ALPHA_ST].setText("");
                inputValues[OFF_ZLT].setText("");
                inputValues[MAT_NX].setText("");
                inputValues[MAT_NY].setText("");
                inputValues[DIS_DX].setText("");
                inputValues[DIS_DY].setText("");
                inputValues[NOB_GOB].setText("");
                inputValues[NUM_CON].setText("");
                inputValues[AREA_FE].setText("");
                inputValues[NUM_CON1].setText("");

 }
}
```

```
package com.polymod3d.view;

import java.awt.*;
import java.io.IOException;

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

import com.polymod3d.model.POLYMOD3D_CalculateValues;


public class POLYMOD3D_TableView extends Panel{

	public static  void populateEastPanel(Object rowData[][]) {
		POLYMOD3D_MainPanel.p_East.removeAll();
		POLYMOD3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYMOD3D_MainPanel.p_East.add("thePlot",POLYMOD3D_MainPanel.thePlot);

		POLYMOD3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYMOD3D_MainPanel.p_East.add("thePlot1",
POLYMOD3D_MainPanel.thePlot1);

		POLYMOD3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYMOD3D_MainPanel.p_East.add("thePlot2",
POLYMOD3D_MainPanel.thePlot2);

	 Object columnNames[] = {"Distance(km)", "Observed anamolies (mGal)",
"Calculated anamolies (mGal)", "Depth(km)"};
		JTable table = new JTable(rowData, columnNames);
		table.setPreferredScrollableViewportSize(new Dimension(300,550));
		JScrollPane scrollPane = new JScrollPane(table);
		scrollPane.setAutoscrolls(true);
		POLYMOD3D_MainPanel.p_East.add(scrollPane);
		POLYMOD3D_MainPanel.p_East.validate();
		POLYMOD3D_MainPanel.p_East.setVisible(true);

}

 public static void DrawTheContourPlot(String obs) {

	String		s;

	try {

		s = obs;
			POLYMOD3D_MainPanel.thePlot.getName("Observed anomaly");
			POLYMOD3D_MainPanel.thePlot.setColor(Color.RED);

POLYMOD3D_MainPanel.thePlot.passObj(POLYMOD3D_MainPanel.thePlot);

POLYMOD3D_MainPanel.thePlot.ParseZedMatrix(s,POLYMOD3D_CalculateValues.con);

POLYMOD3D_MainPanel.thePlot.paint(POLYMOD3D_MainPanel.thePlot.getGraphics(),POLYMOD
3D_CalculateValues.con);

	}
	catch(ParseMatrixException e) {
			POLYMOD3D_MainPanel.thePlot.repaint();
```

```
		}
		catch(IOException e) {
				POLYMOD3D_MainPanel.thePlot.repaint();

		}
		finally {
			//System.out.println("Exiting DrawTheContourPlot");
		}
}
 public static void DrawTheContourPlot1(String obs) {

		String		s;

		try {
			s = obs;
				POLYMOD3D_MainPanel.thePlot1.getName("Modeled anomaly");
				POLYMOD3D_MainPanel.thePlot1.setColor(Color.MAGENTA);

POLYMOD3D_MainPanel.thePlot1.passObj(POLYMOD3D_MainPanel.thePlot1);

POLYMOD3D_MainPanel.thePlot1.ParseZedMatrix(s,POLYMOD3D_CalculateValues.con);

POLYMOD3D_MainPanel.thePlot1.paint(POLYMOD3D_MainPanel.thePlot1.getGraphics(),POLYM
OD3D_CalculateValues.con);
		}
		catch(ParseMatrixException e) {
				POLYMOD3D_MainPanel.thePlot1.repaint();

		}
		catch(IOException e) {
				POLYMOD3D_MainPanel.thePlot1.repaint();

		}
		finally {
			//System.out.println("Exiting DrawTheContourPlot");
		}
}

 public static void DrawTheContourPlot2(String obs) {

		String		s;

		try {
			s = obs;
				POLYMOD3D_MainPanel.thePlot2.getName("Estimated Depth");
				POLYMOD3D_MainPanel.thePlot2.setColor(Color.blue.brighter());

POLYMOD3D_MainPanel.thePlot2.passObj(POLYMOD3D_MainPanel.thePlot2);

POLYMOD3D_MainPanel.thePlot2.ParseZedMatrix(s,POLYMOD3D_CalculateValues.con);

POLYMOD3D_MainPanel.thePlot2.paint(POLYMOD3D_MainPanel.thePlot2.getGraphics(),POLYM
OD3D_CalculateValues.con);

POLYMOD3D_MainPanel.thePlot2.index(POLYMOD3D_MainPanel.thePlot2.getGraphics());
		}
		catch(ParseMatrixException e) {
				POLYMOD3D_MainPanel.thePlot2.repaint();

		}
```

```
		catch(IOException e) {
				POLYMOD3D_MainPanel.thePlot2.repaint();

		}
		finally {
			//System.out.println("Exiting DrawTheContourPlot");
		}
	}
}
```

```
package com.polymod3d.view;

import java.awt.*;
import java.io.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.awt.image.ImageObserver;
import javax.imageio.ImageIO;

import com.polymod3d.model.POLYMOD3D_CalculateValues;


class ContourPlot extends Canvas {


 final static boolean		SHOW_NUMBERS	= true;
 final static int		BLANK		= 32,
				OPEN_SUITE		= (int)'{',
				CLOSE_SUITE		= (int)'}',
				BETWEEN_ARGS		= (int)',',
				N_CONTOURS=10,
				PLOT_MARGIN		= 20,
				WEE_BIT			=  3,
				NUMBER_LENGTH		=  3;
 final static double		Z_MAX_MAX	= 1.0E+10,
				Z_MIN_MIN		= -Z_MAX_MAX;
 final static String EOL		=
				System.getProperty("line.separator");


 int			xSteps, ySteps;
 float			z[][];
 boolean			logInterpolation = false;
 Dimension		d;
 double			deltaX, deltaY;
 String Name ="";
 Color col;
 ImageObserver io;


 int		l1[] = new int[4];
 int		l2[] = new int[4];
 int		ij[] = new int[2];
 int		i1[] = new int[2];
 int		i2[] = new int[2];
 int		i3[] = new int[6];
 int		ibkey,icur,jcur,ii,jj,elle,ix,iedge,iflag,ni,ks;
```

```
int             cntrIndex,prevIndex;
int             idir,nxidir,k;
double          z1,z2,cval,zMax,zMin;
double          intersect[] = new double[4];
double          xy[]       = new double[2];
double          prevXY[] = new double[2];

boolean          jump;


public ContourPlot(int x, int y) {
        super();
        xSteps = x;
        ySteps = y;

                setForeground(Color.black);
                setBackground(Color.white);
}

public void getName(String s){
        Name = s;
}

int          ncv = N_CONTOURS;
float          cv[];
public Graphics g1;

int sign(int a, int b) {
        a = Math.abs(a);
        if (b < 0)                  return -a;
        else                        return  a;
}


void InvalidData() {
        cv[0] = (float)0.0;
        cv[1] = (float)0.0;
}


        void GetExtremes() throws ParseMatrixException {
        int          i,j;
        double          here;

        zMin = z[0][0];
        zMax = zMin;
        for (j = 0; j < ySteps; j++) {
                for (i = 0; i < xSteps; i++) {
                                here = z[i][j];
                                if (zMin > here) zMin = here;
                                if (zMax < here) zMax = here;
                }
        }
        if (zMin == zMax) {
                InvalidData();
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errEqual);
        }
                return;
```

```
}

        void AssignContourValues() throws ParseMatrixException {
         int          i;
         double          delta;

         if ((logInterpolation) && (zMin <= 0.0)) {
                 InvalidData();
                         throw new ParseMatrixException(POLYMOD3D_MainPanel.errLog);
         }
         if (logInterpolation) {
                 double          temp = Math.log(zMin);

                 delta = (Math.log(zMax)-temp) / ncv;
                 for (i = 0; i < ncv; i++) cv[i] = (float)Math.exp(temp + (i+1)*delta);
         }
         else {
                 delta = (zMax-zMin) / ncv;
                 for (i = 0; i < ncv; i++) cv[i] = (float)(zMin + (i+1)*delta);
         }
}


        String GetContourValuesString() {
         String          s = new String();
         int          i;

                for (i = 0; i < ncv; i++) s = s + "[" + Integer.toString(i) + "] " +
                         Float.toString(cv[i]) + EOL;
                return s;
}


        void SetMeasurements() {
         d = size();
         d.width  = 250;//d.width  - 2*PLOT_MARGIN;
         d.height = 250;//d.height - 2*PLOT_MARGIN;
         deltaY = d.width  / (ySteps - 1.0);
         deltaX = d.height / (xSteps - 1.0);
}

void setColor(Color s){

        col = s;
}

void DrawGrid(Graphics2D g) {
        int i,j,kx,ky;
        g.clearRect(0, 0, d.height+2*PLOT_MARGIN+10, d.width +2*PLOT_MARGIN);
        Color c = new Color(0.933f,0.914f,0.749f);
        g.setColor(c);
        g.fill(new Rectangle2D.Float( 50,20,250,250));
                g.setColor(Color.BLACK);
        for (i = 0; i < xSteps; i++) {
                 if(i==0||i==xSteps-1)
                 {
                         kx = (int)((float)i * deltaX);
                         g.drawLine( 30+PLOT_MARGIN,
                                         PLOT_MARGIN+kx,
```

```
                                        30+PLOT_MARGIN+d.height,
                                        PLOT_MARGIN+kx);
                }
        }
        g.drawLine( 30,290,300,290);
        g.drawLine( 30,290,30,20);
        float maxX = (float)
((POLYMOD3D_CalculateValues.input_mat_nx-1)*POLYMOD3D_CalculateValues.input_dx);
        float maxY = (float)
((POLYMOD3D_CalculateValues.input_mat_ny-1)*POLYMOD3D_CalculateValues.input_dy);
                int x = (int) POLYMOD3D_CalculateValues.input_dx;
                int y = (int) POLYMOD3D_CalculateValues.input_dy;
        int k=0;
        int l=0;
        while(x<=maxX){

                g.drawString("'",PLOT_MARGIN+30+250* x/maxX , 297);
                if(k%2==0)
                        g.drawString(""+x,PLOT_MARGIN+27+250* x/maxX, 310);

                        x =(int) (x+POLYMOD3D_CalculateValues.input_dx);
                k=k+1;
        }

        while(y<=maxY){

                g.drawString("-", 30,300-(250*y/maxY)-PLOT_MARGIN);
                if(l%2==0)
                                g.drawString(""+y,15,300-(250*y/maxY)-PLOT_MARGIN);

                        y =(int) (y+POLYMOD3D_CalculateValues.input_dy);
                l=l+1;
        }

        for (i = 0; i < xSteps; i++) {

                kx = (int)((float)i * deltaX);

        }

        for (j = 0; j < ySteps; j++) {
                if(j==0||j==ySteps-1)
                {
                        ky = (int)((float)j * deltaY);
                                g.drawLine(30+ PLOT_MARGIN+ky,
                                        PLOT_MARGIN,
                                                30+PLOT_MARGIN+ky,
                                        PLOT_MARGIN+d.width);
                }
        }


        g.setColor(col);
        g.setFont(new Font("Arial", 20, 20));
        g.drawString(""+Name,70,350);
        g.setFont(new Font("Arial", 20, 12));

}

 void SetColour(Graphics g) {
```

```
        Color c = new Color(
                                ((ncv-cntrIndex) * Color.blue.getRed()   +
                                        cntrIndex * Color.red.getRed())/ncv,
                                ((ncv-cntrIndex) * Color.blue.getGreen() +
                                        cntrIndex * Color.red.getGreen())/ncv,
                                ((ncv-cntrIndex) * Color.blue.getBlue()  +
                                        cntrIndex * Color.red.getBlue())/ncv);

        g.setColor(c);
}

        void DrawKernel(Graphics2D g2) {
        float           prevU,prevV,u,v;

        //DecimalFormat df =new DecimalFormat("0.#");
        if ((iflag == 1) || (iflag == 4) || (iflag == 5)) {
                        if (cntrIndex != prevIndex) { // Must change colour

                                prevIndex = cntrIndex;
                }
                        prevU =(float)((prevXY[0] - 1.0) * deltaX);
                        prevV =(float) ((prevXY[1] - 1.0) * deltaY);
                u = (float)((xy[0] - 1.0) * deltaX);
                v = (float)((xy[1] - 1.0) * deltaY);

                SetColour(g2);
                if(cntrIndex==0){
                        g2.draw(new
Line2D.Float(30+PLOT_MARGIN+prevV,PLOT_MARGIN+prevU,30+PLOT_MARGIN+v,
PLOT_MARGIN+u));
                }
                g2.draw(new
Line2D.Float(30+PLOT_MARGIN+prevV,PLOT_MARGIN+prevU,30+PLOT_MARGIN+v,
PLOT_MARGIN+u));
                if ((SHOW_NUMBERS) && ((iflag==4) || (iflag==5))) {
                                g2.setColor(Color.black);
                        if      (u == 0)            u = u - WEE_BIT;
                        else if         (u == d.width)  u = u + PLOT_MARGIN/2;
                        else if         (v == 0)  v = v - PLOT_MARGIN/2;
                        else if          (v == d.height) v = v + WEE_BIT;
                        if(cntrIndex%3==0)

g2.drawString(""+POLYMOD3D_CalculateValues.df8.format(cv[cntrIndex]),
                                                PLOT_MARGIN+v+30, PLOT_MARGIN+u);

                }

        }
                prevXY[0] = xy[0];
                prevXY[1] = xy[1];
}

 void DetectBoundary() {
        ix = 1;
        if (ij[1-elle] != 1) {
                ii = ij[0] - i1[1-elle];
                jj = ij[1] - i1[elle];
                if (z[ii-1][jj-1] <= Z_MAX_MAX) {
                        ii = ij[0] + i2[elle];
                        jj = ij[1] + i2[1-elle];
```

```
                        if (z[ii-1][jj-1] < Z_MAX_MAX) ix = 0;
                }
                if (ij[1-elle] >= l1[1-elle]) {
                        ix = ix + 2;
                                return;
                }
        }
        ii = ij[0] + i1[1-elle];
        jj = ij[1] + i1[elle];
        if (z[ii-1][jj-1] > Z_MAX_MAX) {
                ix = ix + 2;
                        return;
        }
        if (z[ij[0]][ij[1]] >= Z_MAX_MAX) ix = ix + 2;
}

boolean Routine_label_020() {
        l2[0] =  ij[0];
        l2[1] =  ij[1];
        l2[2] = -ij[0];
        l2[3] = -ij[1];
        idir = 0;
        nxidir = 1;
        k = 1;
        ij[0] = Math.abs(ij[0]);
        ij[1] = Math.abs(ij[1]);
        if (z[ij[0]-1][ij[1]-1] > Z_MAX_MAX) {
                elle = idir % 2;
                ij[elle] = sign(ij[elle],l1[k-1]);
                        return true;
        }
        elle = 0;
                return false;
}

boolean Routine_label_050() {
        while (true) {
                if (ij[elle] >= l1[elle]) {
                        if (++elle <= 1) continue;
                        elle = idir % 2;
                        ij[elle] = sign(ij[elle],l1[k-1]);
                                if (Routine_label_150()) return true;
                        continue;
                }
                ii = ij[0] + i1[elle];
                jj = ij[1] + i1[1-elle];
                if (z[ii-1][jj-1] > Z_MAX_MAX) {
                        if (++elle <= 1) continue;
                        elle = idir % 2;
                        ij[elle] = sign(ij[elle],l1[k-1]);
                                if (Routine_label_150()) return true;
                        continue;
                }
                        break;
        }
        jump = false;
                return false;
}
boolean Routine_label_150() {
        while (true) {
```

```
                if (ij[elle] < l1[k-1]) {
                        ij[elle]++;
                        if (ij[elle] > l2[k-1]) {
                                l2[k-1] = ij[elle];
                                idir = nxidir;
                                nxidir = idir + 1;
                                k = nxidir;
                                if (nxidir > 3) nxidir = 0;
                        }
                        ij[0] = Math.abs(ij[0]);
                        ij[1] = Math.abs(ij[1]);
                        if (z[ij[0]-1][ij[1]-1] > Z_MAX_MAX) {
                                elle = idir % 2;
                                ij[elle] = sign(ij[elle],l1[k-1]);
                                continue;
                        }
                        elle = 0;
                                return false;
                }
                if (idir != nxidir) {
                        nxidir++;
                        ij[elle] = l1[k-1];
                        k = nxidir;
                        elle = 1 - elle;
                        ij[elle] = l2[k-1];
                        if (nxidir > 3) nxidir = 0;
                        continue;
                }

                        if (ibkey != 0) return true;
                ibkey = 1;
                ij[0] = icur;
                ij[1] = jcur;
                if (Routine_label_020()) continue;
                        return false;
        }
}

short Routine_label_200(Graphics g,  boolean workSpace[])
{
        Graphics2D g2 = (Graphics2D) g;
        while (true) {
                xy[elle] = 1.0*ij[elle] + intersect[iedge-1];
                xy[1-elle] = 1.0*ij[1-elle];
                workSpace[2*(xSteps*(ySteps*cntrIndex+ij[1]-1)
                                +ij[0]-1) + elle] = true;
                        DrawKernel(g2);
                if (iflag >= 4) {
                        icur = ij[0];
                        jcur = ij[1];
                                return 1;
                }
                ContinueContour();
                if (!workSpace[2*(xSteps*(ySteps*cntrIndex
                                        +ij[1]-1)+ij[0]-1)+elle]) return 2;
                iflag = 5;                  // 5. Finish a closed contour
                iedge = ks + 2;
                if (iedge > 4) iedge = iedge - 4;
                intersect[iedge-1] = intersect[ks-1];
        }
```

```
}
        boolean CrossedByContour(boolean workSpace[]) {
         ii = ij[0] + i1[elle];
         jj = ij[1] + i1[1-elle];
         z1 = z[ij[0]-1][ij[1]-1];
         z2 = z[ii-1][jj-1];
         for (cntrIndex = 0; cntrIndex < ncv; cntrIndex++) {
                  int i = 2*(xSteps*(ySteps*cntrIndex+ij[1]-1) + ij[0]-1) + elle;

                  if (!workSpace[i]) {
                           float x = cv[cntrIndex];
                           if ((x>Math.min(z1,z2)) && (x<=Math.max(z1,z2))) {
                                    workSpace[i] = true;
                                             return true;
                           }
                  }
         }
                 return false;
}
void ContinueContour() {
         short local_k;

         ni = 1;
         if (iedge >= 3) {
                  ij[0] = ij[0] - i3[iedge-1];
                  ij[1] = ij[1] - i3[iedge+1];
         }
         for (local_k = 1; local_k < 5; local_k++)
                  if (local_k != iedge) {
                           ii = ij[0] + i3[local_k-1];
                           jj = ij[1] + i3[local_k];
                           z1 = z[ii-1][jj-1];
                           ii = ij[0] + i3[local_k];
                           jj = ij[1] + i3[local_k+1];
                           z2 = z[ii-1][jj-1];
                           if ((cval > Math.min(z1,z2) && (cval <= Math.max(z1,z2)))) {
                                    if ((local_k == 1) || (local_k == 4)) {
                                             double          zz = z2;

                                             z2 = z1;
                                             z1 = zz;
                                    }
                                    intersect[local_k-1] = (cval - z1)/(z2 - z1);
                                    ni++;
                                    ks = local_k;
                           }
                  }
         if (ni != 2) {
                  ks = 5 - iedge;
                  if (intersect[2] >= intersect[0]) {
                           ks = 3 - iedge;
                           if (ks <= 0) ks = ks + 4;
                  }
         }
         elle = ks - 1;
         iflag = 1;                 // 1. Continue a contour
         jump = true;
         if (ks >= 3) {
                  ij[0] = ij[0] + i3[ks-1];
                  ij[1] = ij[1] + i3[ks+1];
```

```
                elle = ks - 3;
        }
}
        void ContourPlotKernel(Graphics2D g, boolean workSpace[])
{
        Graphics2D g2 = (Graphics2D) g;
        short val_label_200;

        l1[0] = xSteps;         l1[1] = ySteps;
        l1[2] = -1;l1[3] = -1;
        i1[0] =         1; i1[1] =  0;
        i2[0] =         1; i2[1] = -1;
        i3[0] =         1; i3[1] =  0; i3[2] = 0;
        i3[3] =         1; i3[4] =  1; i3[5] = 0;
                prevXY[0] = 0.0; prevXY[1] = 0.0;
        xy[0] = 1.0; xy[1] = 1.0;
        cntrIndex = 0;
                prevIndex = -1;
        iflag = 6;
                DrawKernel(g2);
        icur = Math.max(1, Math.min((int)Math.floor(xy[0]), xSteps));
        jcur = Math.max(1, Math.min((int)Math.floor(xy[1]), ySteps));
        ibkey = 0;
        ij[0] = icur;
        ij[1] = jcur;
        if (Routine_label_020() &&
                                Routine_label_150()) return;
                if (Routine_label_050()) return;
        while (true) {
                 DetectBoundary();
                 if (jump) {
                        if (ix != 0) iflag = 4; // Finish contour at boundary
                        iedge = ks + 2;
                        if (iedge > 4) iedge = iedge - 4;
                        intersect[iedge-1] = intersect[ks-1];
                        val_label_200 = Routine_label_200(g,workSpace);
                        if (val_label_200 == 1) {
                                        if (Routine_label_020() && Routine_label_150()) return;
                                        if (Routine_label_050()) return;
                                 continue;
                        }
                        if (val_label_200 == 2) continue;
                                return;
                }
                       if ((ix != 3) && (ix+ibkey != 0) && CrossedByContour(workSpace)) {
                        iedge = elle + 1;
                        cval = cv[cntrIndex];
                        if (ix != 1) iedge = iedge + 2;
                        iflag = 2 + ibkey;
                        intersect[iedge-1] = (cval - z1) / (z2 - z1);
                        val_label_200 = Routine_label_200(g,workSpace);
                        if (val_label_200 == 1) {
                                        if (Routine_label_020() && Routine_label_150()) return;
                                        if (Routine_label_050()) return;
                                 continue;
                        }
                        if (val_label_200 == 2) continue;
                                return;
                }
                if (++elle > 1) {
```

```
                        elle = idir % 2;
                        ij[elle] = sign(ij[elle],l1[k-1]);
                                if (Routine_label_150()) return;
                }
                        if (Routine_label_050()) return;
        }
}

public void passObj(ImageObserver s){

        io =s;
}
public void paint(Graphics g,int n)
{
        ncv=n;
        int workLength = 2 * xSteps * ySteps * ncv;
        boolean          workSpace[]; // Allocate below if data valid

                SetMeasurements();
        try {
                int width = 350;
                int height = 360;
                Buff        eredImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buff         er.createGraphics();

                        g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);

                Graphics2D g2 = (Graphics2D)g1 ;
                        g2.setBackground(Color.WHITE);
                DrawGrid(g2);
                        if (cv[0] != cv[1]) { // Valid data
                         workSpace = new boolean[workLength];
                                ContourPlotKernel(g2, workSpace);
                }

                        FileOutputStream os = new
FileOutputStream(POLYMOD3D_CalculateValues.input_area_name+Name+".jpg");
                ImageIO.write(buff         er, "jpg", os);
                os.close();
                        String path =  POLYMOD3D_CalculateValues.input_area_name+Name
+".jpg";
                Buff        eredImage image = ImageIO.read(new File(path));

                        Graphics g_image = g;//POLYMOD3D_MainPanel.p_East.getGraphics();
                g_image.drawImage(image, -10, 100, image.getWidth(), image.getHeight(),io
);

        }
        catch (Exception e2) {

                        e2.printStackTrace();
        }

}
public void index(Graphics g){

        try {
                int width = 90;
```

```
                int height = 90;
                Buff          eredImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buff          er.createGraphics();

                        g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);

                Graphics2D g2 = (Graphics2D)g1 ;
                        g2.setBackground(Color.WHITE);

                drawIndex(g2);


                        FileOutputStream os = new FileOutputStream("Index.jpg");
                ImageIO.write(buff          er, "jpg", os);
                os.close();
                String path =  "Index.jpg";
                Buff          eredImage image = ImageIO.read(new File(path));

                        Graphics g_image = g;//POLYMOD3D_MainPanel.p_East.getGraphics();
                g_image.drawImage(image, 180, 20, image.getWidth(),
image.getHeight(),io );
        }
        catch (Exception e2) {

                        e2.printStackTrace();
        }



}

 public void drawIndex(Graphics2D g){

                g.setColor(Color.black);

        g.drawLine(10, 70, 70, 70);
        g.drawLine(10, 70, 10,0);
        g.setFont(new Font("Arial", 40,11));
        g.drawString("EAST (km)", 20, 80);
        String a[]={"N","O","R","T","H","(k","m)"};
        for(int e=0;e<a.length;e++)
        {
                g.drawString(""+a[e],0,10+(e*10));
        }
}


 public void ParseZedMatrix(String s,int n)
                        throws ParseMatrixException, IOException
{
        StringBuff          erInputStream i;
                StreamTokenizer         t;
        ncv=n;
        cv= new float[ncv+2];
        i = new StringBuff          erInputStream(s);
                t = new StreamTokenizer(i);

        z = null;  // Junk any existing matrix
```

```
        EatCharacter(t,OPEN_SUITE);
                do ParseRowVector(t);
                while (t.nextToken() == BETWEEN_ARGS);
        if (t.ttype != CLOSE_SUITE) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errExpect+(char)CLOSE_SUITE);
        }
                if (t.nextToken() != StreamTokenizer.TT_EOF) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errEOF);
        }
        MakeMatrixRectangular();
                GetExtremes();
        if (zMax > Z_MAX_MAX) zMax = Z_MAX_MAX;
        if (zMin < Z_MIN_MIN) zMin = Z_MIN_MIN;
                AssignContourValues();
}

        public void ParseRowVector(StreamTokenizer t)
                        throws ParseMatrixException, IOException
{
        if (z == null) z = new float[1][];
        else AddRow();
        EatCharacter(t,OPEN_SUITE);
        do {
                        if (t.nextToken() == StreamTokenizer.TT_NUMBER) {
                         int x = z.length - 1;

                         if (z[x] == null) {
                                  z[x] = new float[1];
                                  z[x][0] = (float)t.nval;
                         }
                         else AddColumn((float)t.nval);
                }
                else {
                         int x = z.length - 1;
                         int y = z[x].length - 1;

                         InvalidData();
                                throw new ParseMatrixException(
                                                POLYMOD3D_MainPanel.errParse + EOL +
                                                POLYMOD3D_MainPanel.errComp + " [" +
                                                Integer.toString(x) + "," +
                                                Integer.toString(y) + "]");
                }
                } while (t.nextToken() == BETWEEN_ARGS);
        if (t.ttype != CLOSE_SUITE) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errExpect+(char)CLOSE_SUITE);
        }
}
        public void AddRow() throws ParseMatrixException {
         int leng = z.length;
         float temp[][];
```

```
                if (leng >= POLYMOD3D_MainPanel.MAX_X_STEPS)
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errBounds);
        temp = new float[leng+1][];
        System.arraycopy(z, 0, temp, 0, leng);
        z = temp;
}
public void AddColumn(float val)
                        throws ParseMatrixException
{
        int i = z.length - 1;
        int leng = z[i].length;
        float temp[];

                if (leng >= POLYMOD3D_MainPanel.MAX_Y_STEPS)
                        throw new ParseMatrixException(
                                        POLYMOD3D_MainPanel.errParse + EOL +
                                        POLYMOD3D_MainPanel.errBounds);
        temp = new float[leng+1];
        System.arraycopy(z[i], 0, temp, 0, leng);
        temp[leng] = val;
        z[i] = temp;
}

public void MakeMatrixRectangular() {
        int             i,y,leng;

        xSteps = z.length;
                ySteps = POLYMOD3D_MainPanel.MIN_Y_STEPS;
        for (i = 0; i < xSteps; i++) {
                y = z[i].length;
                if (ySteps < y) ySteps = y;
        }

        for (i = 0; i < xSteps; i++) {
                leng = z[i].length;
                if (leng < ySteps) {
                        float temp[] = new float[ySteps];

                        System.arraycopy(z[i], 0, temp, 0, leng);
                        while (leng < ySteps) temp[leng++] = 0;
                        z[i] = temp;
                }
        }
}
        public String ReturnZedMatrix() {
        String          s,oneValue;
        int             i,j;

        s = new String(
                                POLYMOD3D_MainPanel.infoStrX + xSteps + EOL +
                                POLYMOD3D_MainPanel.infoStrY + ySteps + EOL);
        for (i = 0; i < xSteps; i++) {
                for (j = 0; j < ySteps; j++) {
                                oneValue = Double.toString(z[i][j]);
                                while (oneValue.length() < NUMBER_LENGTH)
                                        oneValue = " " + oneValue;
                                s = s + oneValue;
```

```
                        if (j < ySteps-1) s = s + " ";
                    }
                    s = s + EOL;
            }
                    return s;
    }
            public void EatCharacter(StreamTokenizer t, int c)
                        throws ParseMatrixException, IOException
    {
                    while (t.nextToken() == BLANK) ;
             if (t.ttype != c) {
                     InvalidData();
                        throw new ParseMatrixException(
                                    POLYMOD3D_MainPanel.errParse + EOL +
                                    POLYMOD3D_MainPanel.errExpect + (char)c);
            }
    }
}

class ContourPlotLayout extends java.lang.Object
{

            private static final int COUNT = POLYMOD3D_MainPanel.NUMBER_COMPONENTS;
     private static final int
     MARGIN                =   5,
     MIN_PLOT_DIMEN        = 300,
     LEFT_WIDTH            = 250,
     CBOX_WIDTH            = 130,
     BUTTON_H_POS          = MARGIN + CBOX_WIDTH + MARGIN,
     BUTTON_WIDTH          = LEFT_WIDTH - CBOX_WIDTH - MARGIN,
     LINE_HEIGHT           =  25,
            DATA_HEIGHT = 105,
     MIN_RES_HEIGHT        =  50,
            DATA_V_POS  = MARGIN + MARGIN + LINE_HEIGHT,
     BUTTON_V_POS          = DATA_V_POS + MARGIN + DATA_HEIGHT,
            RESULTS_V_POS       = BUTTON_V_POS + MARGIN + LINE_HEIGHT;
     Component k[]          = new Component[COUNT];
     Dimension d            = new Dimension(MIN_PLOT_DIMEN, MIN_PLOT_DIMEN);
            int results_height        = MIN_RES_HEIGHT;

            public void GetDimensions(Container  parent) {
                    d = parent.size();
             d.width = d.width - LEFT_WIDTH - 3*MARGIN;
             d.height = d.height - 2*MARGIN;
             if (d.width < MIN_PLOT_DIMEN) d.width = MIN_PLOT_DIMEN;
             if (d.height < MIN_PLOT_DIMEN) d.height = MIN_PLOT_DIMEN;
             if (d.width > d.height) d.width = d.height;
             else if (d.height > d.width) d.height = d.width;
                    results_height = d.height + MARGIN - RESULTS_V_POS;
                    if (results_height < MIN_RES_HEIGHT) results_height = MIN_RES_HEIGHT;
     }
     public void addComponentNumber(int i, Component c) {
            if ((i < 0) || (i >= COUNT)) {
                        throw new ArrayIndexOutOfBoundsException();
            }
            else if (k[i] != null) {
                    //                  throw new SomeKindOfException(
                    //                          "Attempt to add a component already added");
            }
            else k[i] = c;
```

```
}

 public void layoutContainer(Container                  parent) {
                GetDimensions(parent);
                if (k[0] != null) k[0].reshape
           (2*MARGIN+LEFT_WIDTH,MARGIN,d.width,d.height);
                if (k[1] != null) k[1].reshape
           (MARGIN,MARGIN,LEFT_WIDTH,LINE_HEIGHT);
                if (k[2] != null) k[2].reshape
                (MARGIN,DATA_V_POS,LEFT_WIDTH,DATA_HEIGHT);
                if (k[3] !=null) k[3].reshape
           (MARGIN,BUTTON_V_POS,CBOX_WIDTH,LINE_HEIGHT);
                if (k[4] != null) k[4].reshape
           (BUTTON_H_POS,BUTTON_V_POS, BUTTON_WIDTH,LINE_HEIGHT);
                if (k[5] != null) k[5].reshape
                (MARGIN,RESULTS_V_POS,LEFT_WIDTH,results_height);
 }

          public Dimension minimumLayoutSize(Container  parent) {
                return new Dimension(
                        3*MARGIN + LEFT_WIDTH + MIN_PLOT_DIMEN,
                        2*MARGIN + MIN_PLOT_DIMEN);
 }

          public Dimension preferredLayoutSize(Container     parent) {
                GetDimensions(parent);
                return new Dimension(3*MARGIN + d.width + LEFT_WIDTH,
                        2*MARGIN + d.height);
 }

          public void removeLayoutComponent(Component c) {
                for (int i = 0; i < COUNT; i++) if (c == k[i]) k[i] = null;
 }
}

class ParseMatrixException extends Exception {
 public ParseMatrixException(String message) { super(message); }
}
```

```
package com.polymod3d.model;

import java.text.DecimalFormat;
import java.util.HashMap;


import com.polymod3d.util.POLYMOD3D_Utility;
import com.polymod3d.view.POLYMOD3D_TableView;

public class POLYMOD3D_CalculateValues {

 public static Object obj[][] = null;
 public static double [][]gobse;
 public static double [][]gcalu;
 public static double [][]zdep;

 public static int input_n_obs = 0 ;
 public static int input_mat_nx = 0;
 public static int input_mat_ny = 0;
```

```
public static double input_dx = 0;
public static double input_dy = 0;
public static int input_num_con_gr = 0;
public static int input_num_con_dep = 0;
double input_sd_poly=0;
double input_almda_st=0;

double input_zlt_km = 0;
final String EOL           =
                        System.getProperty("line.separator");

        public static String input_area_name ,input_pro_name= "";
public static String val=null;
public static String val1=null;
public static String val2=null;
public static int con = 0;
public static DecimalFormat df8;

        public void getAnamolyValues(HashMap h_Map) {

         double input_nob_gob[] = null;



         try {

                  input_n_obs =
POLYMOD3D_Utility.convertInteger((String)h_Map.get("N_OBS"));
                  input_nob_gob =
POLYMOD3D_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                  input_sd_poly =
POLYMOD3D_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                  input_almda_st =
POLYMOD3D_Utility.convertDouble((String)h_Map.get("ALPHA_ST"));;
                  input_mat_nx =
POLYMOD3D_Utility.convertInteger((String)h_Map.get("MAT_NX"));
                  input_mat_ny =
POLYMOD3D_Utility.convertInteger((String)h_Map.get("MAT_NY"));
                  input_dx =
POLYMOD3D_Utility.convertDouble((String)h_Map.get("DIS_DX"));
                  input_dy =
POLYMOD3D_Utility.convertDouble((String)h_Map.get("DIS_DY"));
                  input_zlt_km =
POLYMOD3D_Utility.convertDouble((String)h_Map.get("OFF_ZLT"));
                  input_num_con_gr =
POLYMOD3D_Utility.convertInteger((String)h_Map.get("NUM_CON"));
                  input_num_con_dep =
POLYMOD3D_Utility.convertInteger((String)h_Map.get("NUM_CON1"));
                        input_area_name =
POLYMOD3D_Utility.convertString((String)h_Map.get("AREA_FE"));

         }
         catch(Exception e) {
                        e.printStackTrace();
         }
         zdep = new double[input_mat_ny+2][input_mat_nx+2];
         gcalu = new double[input_mat_ny+2][input_mat_nx+2];

         gobse = new double[input_mat_ny+2][input_mat_nx+2];
```

```
double conv[][] = new double[input_mat_ny+1][input_mat_nx+1];


double a[]=new double[input_mat_nx*input_mat_ny+1];
for (int i = 1;i <= input_mat_nx * input_mat_ny; i++) {
        a[i - 1] = input_nob_gob[i];
}
for (int I = 0;I < input_mat_ny; I++) {
        for (int K = 0;K < input_mat_nx; K++) {
                conv[I][K] = a[(I * input_mat_nx) + K];
        }
}
for (int I = 1;I <= input_mat_ny; I++) {
        for (int K = 1;K <= input_mat_nx; K++) {
                gobse[I][K]= conv[I-1][K-1];
        }

}


double x[] = new double[input_mat_nx+1];
double y[] = new double[input_mat_ny+1];
double xx[] = new double[input_mat_nx+2];
double yy[] = new double[input_mat_ny+2];

double err[][] = new double[input_mat_ny+2][input_mat_nx+2];
double z1[][] = new double[input_mat_ny+2][input_mat_nx+2];
double gc1[][] = new double[input_mat_ny+2][input_mat_nx+2];
double dcz[][] = new double[input_mat_ny+1][input_mat_nx+1];


double pi = 3.14159265;
double tog = 13.3333;
double aer = 0.0001*input_mat_ny*input_mat_nx;
for (int i = 1;i <= input_mat_nx; i++) {
        x[i] = (i - 1)*input_dx;
}
for (int i = 1;i <= input_mat_ny; i++) {
        y[i] = (i - 1)*input_dy;
}
```

```
	for (int jj = 2; jj <= input_mat_ny-1; jj++) {

		for (int kk = 2; kk <= input_mat_nx-1; kk++) {

			if(input_almda_st == 0)
				zdep[jj][kk] = gobse[jj][kk] / (tog * pi * input_sd_poly);
			else
				zdep[jj][kk] = -(1 / input_almda_st) * Math.log(1 -
((input_almda_st * gobse[jj][kk]) / (tog * pi * input_sd_poly)));


		}
	}


	for(int ITER = 1; ITER <= input_n_obs; ITER++) {

		double ITER1 = (ITER+1)-1;
		double   MM = input_n_obs;

		System.out.println(ITER1);

		for (int IY = 1; IY <= input_mat_ny; IY++) {

			for (int IX = 1;IX <= input_mat_nx; IX++) {

				gc1[IY][IX] = 0;
				z1[IY][IX]=0;

			}
		}

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

			for (int i = 1;i <= input_mat_nx; i++) {

				for(int iy = 1;iy <=input_mat_ny; iy++) {

					yy[iy] = y[iy]-y[k];


				}
				for(int ix=1;ix<=input_mat_nx;ix++) {

					xx[ix] = x[ix] - x[i];


				}


					gc1[k][i] = g3DBasin(xx, zdep, yy, input_mat_nx,
input_mat_ny, input_sd_poly, input_almda_st);

			}
		}
		double func = 0;
		for (int l = 1; l <= input_mat_ny; l++) {

			for (int m = 1; m <= input_mat_nx; m++) {
```

```
                        func = func+ Math.pow(gobse[l][m]- gc1[l][m], 2);

                }
        }


        System.out.println(func);



        for (int i = 2;i <= input_mat_ny-1; i++) {

                for (int k = 2;k <= input_mat_nx-1; k++) {
                        err[i][k] = gobse[i][k] - gc1[i][k];
                        double st = tog * pi * input_sd_poly * Math.exp(-
input_almda_st * zdep[i][k]);
                        dcz[i][k] = -(1 / input_almda_st) * Math.log(1-((input_almda_st
* err[i][k]) / st));
                        z1[i][k] = zdep[i][k] + dcz[i][k];
                        if(z1[i][k] < 0) z1[i][k] = 0.0;
                        if(z1[i][k] > input_zlt_km) z1[i][k] =input_zlt_km;
                }
        }


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

                for (int i = 1;i <= input_mat_nx; i++) {

                        for(int iy = 1;iy <= input_mat_ny; iy++) {

                                yy[iy] = y[iy]-y[k];

                        }
                        for(int ix=1;ix<=input_mat_nx;ix++) {

                                xx[ix] = x[ix] - x[i];

                        }


                                gcalu[k][i] = g3DBasin(xx, z1, yy, input_mat_nx, input_mat_ny,
input_sd_poly, input_almda_st);

                }
        }
        double func1 = 0;
        for (int l = 1;l <= input_mat_ny; l++) {

                for (int m = 1;m <= input_mat_nx; m++) {

                        func1 = func1 + Math.pow(gobse[l][m] - gcalu[l][m], 2);

                }
        }


        System.out.println(func1);
```

```
if (func1 > func  )
                break;
//funct = funct1;
if (func1 < aer ) {

                break;
}

else  {

        func = func1;

        for (int i = 2;i <= input_mat_ny-1; i++) {

                for (int k = 2;k <= input_mat_nx-1; k++) {

                        zdep[i][k] = z1[i][k];
                }
        }
        //System.out.println(funct);
}



//if (ITER1 == input_n_obs)
{
        String a1 = "{";
        String a21 = "{";
        String a31 = "{";
        String a2 = "}";
        String a4 = "{";
        String a3[] = new String[input_mat_nx+1];

        for(int k = 0; k < input_mat_nx; k++) {

                a3[k]="{";
        }

                double regob[][] = new double[input_mat_ny*5][input_mat_nx*5];
                double regcal[][] = new double[input_mat_ny+1][input_mat_nx+1];
                double rezcal[][] = new double[input_mat_ny+1][input_mat_nx+1];
        String b1[][] = new String[input_mat_ny*5][input_mat_nx*5];
        String b2[][] = new String[input_mat_ny+1][input_mat_nx+1];
        String b3[][] = new String[input_mat_ny+1][input_mat_nx+1];

        for(int I=0;I<=input_mat_ny;I++) {



                for(int K = 0;K <= input_mat_nx; K++) {


                        DecimalFormat df =new DecimalFormat("0.####");


                                regob[I][K] = gobse[ I][K];
                                regcal[I][K] = gcalu[ I][K];
                                rezcal[I][K] = zdep[ I][K];
```

```
                                        b1[I][K] = Double.toString(regob[I][K]);
                                        b2[I][K] = df.format(regcal[I][K]);//
Double.toString(regcal[K][I]);
                                        b3[I][K] = df.format(rezcal[I][K]);//
Double.toString(rezcal[K][I]);

                                if(K < (input_mat_nx )) {

                                        b1[I][K] = b1[I][K] + ",";
                                        b2[I][K] = b2[I][K] + ",";
                                        b3[I][K] = b3[I][K] + ",";
                                }
                                a1 = a1 + b1[I][K];
                                a21 = a21 + b2[I][K];
                                a31 = a31 + b3[I][K];
                        }

                        if (I < (input_mat_ny )) {

                                a1 = a1 + a2 + "," + EOL + a3[I];
                                a21 = a21 + a2 + "," + EOL + a3[I];
                                a31 = a31 + a2 + "," + EOL + a3[I];
                        }
                }
                a1 = a4 + a1 + a2 + a2;
                a21 = a4 + a21 + a2 + a2;
                a31 = a4 + a31 + a2 + a2;
                val = a1;
                val1 = a21;
                val2 = a31;
                a1 = null;
                a21 = null;
                a31 = null;
        }

        con = input_num_con_gr;
        df8 =new DecimalFormat("0");

POLYMOD3D_TableView.DrawTheContourPlot(POLYMOD3D_CalculateValues.val);

POLYMOD3D_TableView.DrawTheContourPlot1(POLYMOD3D_CalculateValues.val1);
        con = input_num_con_dep;
        df8 =new DecimalFormat("0.#");

POLYMOD3D_TableView.DrawTheContourPlot2(POLYMOD3D_CalculateValues.val2);
    }
        setGraphValues(input_n_obs,gobse,gcalu,zdep,input_mat_ny,input_mat_nx);

}



    public double g3DBasin(double []x, double [][]zv, double[]y,int nx, int ny,double sd, double
la) {

    double []gc = new double[ 10000];
    double []zt = new double[10000];
    double []x1 = new double[10000];
    double []gs = new double[10000];
    for(int I = 1; I <= ny; I++){
```

```
double dg = 0.0;
x[nx + 1] = x[1];
zv[l][nx + 1] = zv[l][1];
Kval:for(int K = 1; K <= nx; K++){
	int K1 = K + 1;
	if(zv[l][K] == 0.0 && zv[l][K1] == 0.0)
		continue Kval;
	//	  if(zv[l][K]!=0.0 && zv[l][K1]!=0.0){

	double	dzz = zv[l][K1] - zv[l][K];
	double dx = x[K1] - x[K];
	double r = Math.sqrt(dx * dx + dzz * dzz);
	if(r==0.0)
			break;
	//if(r!=0){
	double c = dx / r;
	double s = dzz / r;
	double cs = 1 / s;
	double ct = c / s;
	double dx1 = Math.abs((x[K1] - x[K])) / 2;

	double zb = (zv[l][K + 1] - zv[l][K]);
	int nd = (int)(zb / dx1) + 1;
	double n1 = nd / 2;
	if(nd-2*n1!=0)//1,2,1
		nd = nd + 1;
	double dz = zb / nd;
	int n2 = nd + 1;
	for(int JZ = 1; JZ <= n2; JZ++){
		zt[JZ] = zv[l][K] + dz * (JZ - 1);
		x1[JZ] = x[K] + (zt[JZ] - zv[l][K]) * ct;
		if(x1[JZ]>0.0 && x1[JZ]<0.0001) x1[JZ] = 0.0;
		if(x1[JZ]<0.0 && x1[JZ]>-0.0001) x1[JZ] = 0.0;

	}
	for(int JZ = 1; JZ <= n2; JZ++){
		double dc = sd * Math.exp(-la * zt[JZ]);
		if (y[l] != 0.0){
			double a = x1[JZ] - zt[JZ] * ct;
			double b = 2.0 * a * ct;
			double anum = (a + zt[JZ] * ct) * zt[JZ];
			double den1 = Math.pow(y[l], 2) + Math.pow(zt[JZ], 2);
			double den2 = Math.sqrt((Math.pow(cs, 2) *
Math.pow(zt[JZ], 2))+(b * zt[JZ]) + Math.pow(a, 2)+Math.pow(y[l], 2));
			double den = den1 * den2;
			double ter = -20.0 / 3.0 * dc * anum / den;
			gs[JZ]= ter;
		}

		else{
			if(x1[JZ] == 0.0 && zt[JZ] == 0.0){
				double anum1 = 0.5 * ct;
				double den1 = Math.sqrt(0.25 +
(Math.pow(zt[JZ], 2) * Math.pow(cs, 2)));
				double ter = -13.3333 * dc *
Math.atan(anum1 / den1);
				gs[JZ]= ter;
			}
			else{
```

```
                                                double a1 = x1[JZ] - zt[JZ] * ct;
                                                double anum2 = 0.5 * (a1 + zt[JZ] * ct);
                                                double den2 = zt[JZ] * Math.sqrt(Math.pow((a1
+ zt[JZ] * ct), 2) + 0.25 + Math.pow(zt[JZ], 2));
                                                double ter1 = -13.3333 * dc *
Math.atan(anum2 / den2);
                                                gs[JZ]= ter1;
                                        }
                                }

                        }
                        double          gr1=0.0;
                        for(int KK1 = 1; KK1 <= n2 - 1; KK1++){
                                double g11 = gs[KK1];
                                double g21 = gs[KK1 + 1];
                                double dzy = zt[KK1 + 1] - zt[KK1];
                                gr1=gr1+dzy*EXPNTR(g11, g21);
                        }
                        dg = dg + gr1;

                }

                gc[l] = dg;

        }
        double gr = 0.0;
        for(int KK = 1; KK <= ny - 1; KK++){
                double g1 = gc[KK];
                double g2 = gc[KK + 1];
                double dy = y[KK + 1] - y[KK];
                gr = gr + dy * EXPNTR(g1, g2);
        }
                double GRAV3D = gr;
                return GRAV3D;

}



 public static double  EXPNTR(double g1, double g2){

        double c;
        double expnt;
        if (g1 == 0)
                g1 = 0.0001;
        if(g2 == 0)
                g2 = 0.0001;
        if( g2 / g1 < 0) {
                double x = Math.abs(g1) / (Math.abs(g1) + Math.abs(g2));
                double  g = 0.00001 * g1 / Math.abs(g1);
                c = Math.log(g / g1);
                double f1 = ( g - g1) / c;
                g = 0.00001 * g2 / Math.abs(g2);
                c = Math.log(g2 / g);
                double f2 = (g2 - g) / c;
                expnt = f1 * x + f2 * (1 - x);
        }
        else {
                c = Math.log(g2 / g1);
```

```
                if(Math.abs(c) <= 0.0001){
                        expnt = g1;
                }
                else {
                        expnt = (g2 - g1) / c;
                }
        }//expnt = dx * expnt;
                return expnt;

}
        public static void  setGraphValues(int i_no_obs,double [][]GOBS,double [][]GCAL,double [
][]dep,int ny,int nx) {

        obj = new Object[nx*ny*5][4];
        int K1;
        DecimalFormat df =new DecimalFormat("0.###");
        for (int i=0; i<= ny;i++) {

                int col = i*nx;
                if(i<ny)
                        obj[col+i][0] = "PROFILE :"+(i+1);
                for (int K = 0;K <= nx;K++){
                        if(K<nx&& i<ny){
                                K1=K+1;
                                obj[col+K+i+1][0] = "" + K1;
                                obj[col+K+i+1][1] = "" + df.format(GOBS[i+1][K+1]);
                                obj[col+K+i+1][2] = "" + df.format(GCAL[i+1][K+1]);
                                obj[col+K+i+1][3] = "" + df.format(dep[i+1][K+1]);
                        }
                }
        }
}
}
```

```
package com.polymod3d.control;

import java.awt.event.*;
import java.io.File;
import java.text.DecimalFormat;

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

import com.polymod3d.model.POLYMOD3D_CalculateValues;
import com.polymod3d.view.POLYMOD3D_MainPanel;
import com.polymod3d.view.POLYMOD3D_TableView;



public class POLYMOD3D_Controller implements ActionListener{


        com.polymod3d.model.POLYMOD3D_CalculateValues cv = new
com.polymod3d.model.POLYMOD3D_CalculateValues();
        Object rowdata[][]={};
 public static boolean success=false;
```

```
 public static boolean success1,success2,success3=false;

 public void actionPerformed(ActionEvent ae) {

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


                        com.polymod3d.view.POLYMOD3D_TableView.populateEastPanel(rowdata);

cv.getAnamolyValues(com.polymod3d.view.POLYMOD3D_MainPanel.captureValues());

com.polymod3d.view.POLYMOD3D_TableView.populateEastPanel(POLYMOD3D_CalculateValues.
obj);
                        POLYMOD3D_MainPanel.p_Center.addMouseListener(new MouseAction());
                        POLYMOD3D_CalculateValues.obj = null;
                        com.polymod3d.view.POLYMOD3D_MainPanel.p_East.repaint();
                        com.polymod3d.view.POLYMOD3D_MainView mv = new
com.polymod3d.view.POLYMOD3D_MainView();
                        mv.setResizable(true);

        }          else if(ae.getActionCommand().equals("Contour")){
                        POLYMOD3D_CalculateValues.con =
POLYMOD3D_CalculateValues.input_num_con_gr;
                        POLYMOD3D_CalculateValues.df8 =new DecimalFormat("0");

POLYMOD3D_TableView.DrawTheContourPlot(POLYMOD3D_CalculateValues.val);

POLYMOD3D_TableView.DrawTheContourPlot1(POLYMOD3D_CalculateValues.val1);
                        POLYMOD3D_CalculateValues.con =
POLYMOD3D_CalculateValues.input_num_con_dep;
                        POLYMOD3D_CalculateValues.df8 =new DecimalFormat("0.#");

POLYMOD3D_TableView.DrawTheContourPlot2(POLYMOD3D_CalculateValues.val2);

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

                try
                {
                                POLYMOD3D_PrintValues.printGraphValues();

                }
                catch(Exception e1) {

                                e1.printStackTrace();
                }

        }else if(ae.getActionCommand().equals("Load data")){
                        POLYMOD3D_MainPanel.loadData();
                        //POLYMOD3D_MainPanel.setDefaultValues();
        }else if(ae.getActionCommand().equals("Clear")){


                        POLYMOD3D_MainPanel.clearDefaultValues();
                        POLYMOD3D_MainPanel.clearPanel(POLYMOD3D_MainPanel.p_Center);
                        com.polymod3d.view.POLYMOD3D_TableView.populateEastPanel(rowdata);
        }
        else if(ae.getActionCommand().equals("Exit")){
                JFrame frame = null;

                int r =         JOptionPane.showConfirmDialog(
```

```
                                        frame,
                                                "Exit POLYMOD3D ?",
                                        "Confirm Exit ",
                                        JOptionPane.YES_NO_OPTION);
                        if(r == JOptionPane.YES_OPTION ){
                                        if(POLYMOD3D_Controller.success==false){
                                        String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Observed anomaly.jpg";
                                        File f = new File(fileName);
                                        f.delete();
                                }
                                        if(POLYMOD3D_Controller.success1==false){
                                        String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Modeled anomaly.jpg";
                                        File f = new File(fileName);
                                        f.delete();
                                }
                                        if(POLYMOD3D_Controller.success2==false){
                                        String fileName =
POLYMOD3D_CalculateValues.input_area_name+"Estimated Depth.jpg";
                                        File f = new File(fileName);
                                        f.delete();
                                }
                                        if(POLYMOD3D_Controller.success3==false){
                                        String fileName = "Index.jpg";
                                        File f = new File(fileName);
                                        f.delete();
                                }
                                System.exit(0);
                        }
                }
        }

}

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

                POLYMOD3D_TableView.DrawTheContourPlot(POLYMOD3D_CalculateValues.val);

POLYMOD3D_TableView.DrawTheContourPlot1(POLYMOD3D_CalculateValues.val1);

POLYMOD3D_TableView.DrawTheContourPlot2(POLYMOD3D_CalculateValues.val2);
 }

}
```

```
package com.polymod3d.control;

import java.io.File;
import java.io.FileWriter;
import java.text.DecimalFormat;
import javax.swing.JFileChooser;
import com.polymod3d.model.POLYMOD3D_CalculateValues;

public class POLYMOD3D_PrintValues {
```

```
        public static void printGraphValues() throws Exception {
         try{
                        String current = System.getProperty("user.dir");
                        File img_file = new File(POLYMOD3D_CalculateValues.input_area_name
+"Observed anomaly.jpg");
                        File img_file1 = new File(POLYMOD3D_CalculateValues.input_area_name
+"Modeled anomaly.jpg");
                        File img_file2 = new File(POLYMOD3D_CalculateValues.input_area_name
+"Estimated Depth.jpg");
                File img_file3 = new File("Index.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());
                                        POLYMOD3D_Controller.success = img_file.renameTo(new
File(dir,img_file.getName()));
                                        POLYMOD3D_Controller.success1 = img_file1.renameTo(new
File(dir,img_file1.getName()));
                                        POLYMOD3D_Controller.success2 = img_file2.renameTo(new
File(dir,img_file2.getName()));
                                        POLYMOD3D_Controller.success3 = img_file3.renameTo(new
File(dir,img_file3.getName()));
                                        myWriter = new FileWriter(OutFile+".html");
                                        myWriter.write(" </table> </td> <td> <img src =
'"+"Index.jpg'");
                                        myWriter.write(" </table> </td> <td> <img src = '"+
POLYMOD3D_CalculateValues.input_area_name +"Observed anomaly.jpg'>"+"<img src = '"+
POLYMOD3D_CalculateValues.input_area_name +"Modeled anomaly.jpg'>"+"<img src = '"+
POLYMOD3D_CalculateValues.input_area_name +"Estimated Depth.jpg'</td></tr></table>");
                                        myWriter.write("<html> <Body onLoad = \"window.print()
\"><table> <tr> <td>" +
                                                        "<table border = 1> <tr> <th colspan =
4>LOCATION:- "+POLYMOD3D_CalculateValues.input_area_name+"</th> </tr>");


                                DecimalFormat df =new DecimalFormat("0.###");
                                        myWriter.write(" <tr><th colspan = 4> ITERATION"+"
"+POLYMOD3D_CalculateValues.input_n_obs+" </th></tr>");

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

                                        for(int i = 1;i <= POLYMOD3D_CalculateValues.input_mat_ny;
i++) {

                                                myWriter.write("<tr><td>PROFILE NO :" + i+"</td></
tr>");
                                         for ( int K = 1; K <=
POLYMOD3D_CalculateValues.input_mat_nx; K++){

                                                        myWriter.write("<tr> <td>" + K+"</td>
<td>"+df.format(POLYMOD3D_CalculateValues.gobse[i][K])+"</td>
<td>"+df.format(POLYMOD3D_CalculateValues.gcalu[i][K])+"</td>
<td>"+df.format(POLYMOD3D_CalculateValues.zdep[i][K])+"</td></tr>");
```

```
                                        }

                                }


                                        myWriter.close();
                        }
                }
                else
                {
                        //pops up error message
                }
        }
        catch(Exception e1) {

                e1.printStackTrace();
        }

}

}
```

```
package com.polymod3d.util;

public class POLYMOD3D_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 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;
 }
}
```

```
package com.polyinv3d.view;

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

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

import com.polyinv3d.control.POLYINV3D_Controller;
import com.polyinv3d.model.POLYINV3D_AnomalyValues;

public class POLYINV3D_MainView  extends Frame {

 /**
          AUTOMATIC INVERSION 3D POLYGON
  *
  */

         private static final long serialVersionUID = 1L;

 public static void main(String s[])
 {
                  POLYINV3D_MainView cm = new POLYINV3D_MainView();
           cm.setSize(1280, 768);
           cm.addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                             JFrame frame = null;

                             int r =          JOptionPane.showConfirmDialog(
                                       frame,
                                                "Exit POLYINV3D MODELLING ?",
                                       "Confirm Exit ",
                                       JOptionPane.YES_NO_OPTION);
                             if(r == JOptionPane.YES_OPTION ){
                                       if(POLYINV3D_Controller.success==false){
                                       String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Observed anomaly.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                  }
                                       if(POLYINV3D_Controller.success1==false){
                                       String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Modeled anomaly.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                  }
                                       if(POLYINV3D_Controller.success2==false){
                                       String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Estimated Depth.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                  }
                                       if(POLYINV3D_Controller.success3==false){
                                       String fileName = "Index.jpg";
                                       File f = new File(fileName);
                                       f.delete();
                                  }
                                  System.exit(0);
```

```
                        }
                }
        });
                cm.setTitle("POLYINV3D");
        cm.setResizable(false);
                cm.add(new POLYINV3D_MainPanel());
                cm.setVisible(true);

}

}
```

```
package com.polyinv3d.view;

import java.awt.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.HashMap;

import javax.swing.JFileChooser;

public class POLYINV3D_MainPanel extends Panel {

 /**
  *
  */
        private static final long serialVersionUID = 1L;
 Panel p_North;
        static Panel p_West;
 public static Panel p_East;
 static Panel p_South;
 public static Panel p_Center;

        public static TextArea img = new TextArea(10,50);
        public static TextArea img1 = new TextArea(10,50);
        public static TextArea img2 = new TextArea(10,50);
        static TextField inputValues [] = new TextField[13];

 Button actionButton[] = new Button[7];
        Object rowdata[][]={};

 final static int NUMBER_COMPONENTS          = 6;
 final static int MIN_X_STEPS =   2,
                  MIN_Y_STEPS =   2,
                  MAX_X_STEPS = 100,
                  MAX_Y_STEPS = 100,
                  N_CONTOURS=10;
 static String            contourValuesTitle,infoStrX,infoStrY,
 errParse,errLog,errComp,errEqual,
        errExpect,errEOF,errBounds;

 public static ContourPlot thePlot              =      new
ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);
 public static ContourPlot thePlot1             =    new
ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);
 static ContourPlot thePlot2            =    new ContourPlot(MIN_X_STEPS,MIN_Y_STEPS);
```

```
final static int AREA_FE = 0;
        final static int MAT_NY = 1;
        final static int MAT_NX = 2;
final static int DIS_DY = 3;
final static int DIS_DX = 4;
        final static int SD_POLY = 5;
final static int ALPHA_ST = 6 ;
        final static int OFF_ZLT = 7;
final static int NOB_GOB = 8;
final static int N_OBS = 9 ;
final static int NUM_CON = 10;
final static int NUM_CON1 = 11;

//final static int PROFILE_NAME = 11;

        public POLYINV3D_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();

         for(int i = 0; i < 13; i++){
                        inputValues[i] = new TextField();
         }
         actionButton[0] = new Button("Inversion");
         actionButton[1] = new Button("Contour");
         actionButton[2] = new Button("Save & Print");
         actionButton[3] = new Button("Load data");
         actionButton[4] = new Button("Clear");
         actionButton[5] = new Button("Exit");

         this.populateNorthPanel();

                POLYINV3D_TableView.populateEastPanel(rowdata);
                this.add(p_North, BorderLayout.NORTH);
                p_Center.setSize(1000, 760);
                this.add(p_Center, BorderLayout.CENTER);
                //this.populateCentre();
                this.add(p_East, BorderLayout.EAST);
                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("No. of profiles along Y axis"));
                p_North.add(inputValues[1]);
         p_North.add(new Label("No. of observations along x axis"));
                p_North.add(inputValues[2]);
                p_North.add(new Label("Profile spacing along Y axis(km)"));
                p_North.add(inputValues[3]);
         p_North.add(new Label("Station spacing along X axis(km)"));
                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 (gm/cc/km)"));
                p_North.add(inputValues[6]);



        p_North.add(new Label("Maximum limiting depth(km)"));
                p_North.add(inputValues[7]);




        p_North.add(new Label("Observed anomalies (mGal)"));
                p_North.add(inputValues[8]);

        p_North.add(new Label("Number of iterations"));
                p_North.add(inputValues[9]);


        p_North.add(new Label("Number of contours for gravity anomalies"));
                p_North.add(inputValues[10]);

        p_North.add(new Label("Number of depth contours "));
                p_North.add(inputValues[11]);

        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.polyinv3d.control.POLYINV3D_Controller());
        actionButton[1].addActionListener(new
com.polyinv3d.control.POLYINV3D_Controller());
        actionButton[2].addActionListener(new
com.polyinv3d.control.POLYINV3D_Controller());
        actionButton[3].addActionListener(new
com.polyinv3d.control.POLYINV3D_Controller());
        actionButton[4].addActionListener(new
com.polyinv3d.control.POLYINV3D_Controller());
        actionButton[5].addActionListener(new
com.polyinv3d.control.POLYINV3D_Controller());

 }

       public static HashMap captureValues(){

        HashMap h_Map = new HashMap();
        try {

                        h_Map.put("AREA_FE",inputValues[AREA_FE].getText());
                        h_Map.put("MAT_NY", inputValues[MAT_NY].getText());
                        h_Map.put("MAT_NX", inputValues[MAT_NX].getText());
                        h_Map.put("DIS_DY", inputValues[DIS_DY].getText());
                        h_Map.put("DIS_DX", inputValues[DIS_DX].getText());
                        h_Map.put("SD_POLY", inputValues[SD_POLY].getText());
```

```
                        h_Map.put("ALPHA_ST",inputValues[ALPHA_ST].getText());
                        h_Map.put("OFF_ZLT", inputValues[OFF_ZLT].getText());
                        h_Map.put("NOB_GOB", inputValues[NOB_GOB].getText());
                        h_Map.put("N_OBS", inputValues[N_OBS].getText());
                        h_Map.put("NUM_CON", inputValues[NUM_CON].getText());
                        h_Map.put("NUM_CON1", inputValues[NUM_CON1].getText());
                        //h_Map.put("PROFILE_NAME",inputValues[PROFILE_NAME].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, 35, 1280, 650);
}



 public static void loadData(){
        try{

                        String current = System.getProperty("user.dir");
                        JFileChooser chooser=new  JFileChooser(current);
                        int returnVal = chooser.showOpenDialog(null);
                 int count = 0;
                        if(returnVal == JFileChooser.APPROVE_OPTION) {
                                File f = chooser.getSelectedFile();
                         Buff          eredReader br=new BufferedReader(new FileReader(f));
                         String st;
                                st = br.readLine();
                         count++;
                         while((st) != null){

                                  try {

                                          if (count == 1){


POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.AREA_FE].setText(""+st);

                                          }

                                          if (count == 2){

POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.MAT_NY].setText(""+st);


                                          }

                                          if (count == 3){


POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.MAT_NX].setText(""+st);
```

```
                }
                if (count == 4){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.DIS_DY].setText(""+st);
                }
                if (count == 5){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.DIS_DX].setText(""+st);
                }

                if (count == 6){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.SD_POLY].setText(""+st);
                }
                if (count == 7){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.ALPHA_ST].setText(""+st);
                }
                if (count == 8){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.OFF_ZLT].setText(""+st);
                }

                if (count == 9){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.NOB_GOB].setText(""+st);
                }
                if (count == 10){
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.N_OBS].setText(""+st);
                }
                if (count == 11){
```

```
POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.NUM_CON].setText(""+st);
                                        }
                                        if (count == 12){

POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.NUM_CON1].setText(""+st);
                                        }
                                        /*          if (count == 12){

POLYINV3D_MainPanel.inputValues[POLYINV3D_MainPanel.PROFILE_NAME].setText(""+st);
                                        }
                                         */

                                }
                                catch(Exception e) {

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

        }
}


        public static void clearDefaultValues(){

                inputValues[N_OBS].setText("");
                inputValues[SD_POLY].setText("");
                inputValues[ALPHA_ST].setText("");
                inputValues[OFF_ZLT].setText("");
                inputValues[MAT_NX].setText("");
                inputValues[MAT_NY].setText("");
                inputValues[DIS_DX].setText("");
                inputValues[DIS_DY].setText("");
                inputValues[NOB_GOB].setText("");
                inputValues[NUM_CON].setText("");
                inputValues[AREA_FE].setText("");
                inputValues[NUM_CON1].setText("");

 }
}
```

```
package com.polyinv3d.view;

import java.awt.*;
```

```
import java.io.IOException;

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

import com.polyinv3d.model.POLYINV3D_AnomalyValues;



public class POLYINV3D_TableView extends Panel{

	public static  void populateEastPanel(Object rowData[][]) {
		POLYINV3D_MainPanel.p_East.removeAll();
		POLYINV3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYINV3D_MainPanel.p_East.add("thePlot",POLYINV3D_MainPanel.thePlot);

		POLYINV3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYINV3D_MainPanel.p_East.add("thePlot1", POLYINV3D_MainPanel.thePlot1);

		POLYINV3D_MainPanel.p_East.setLayout(new GridLayout());
		POLYINV3D_MainPanel.p_East.add("thePlot2", POLYINV3D_MainPanel.thePlot2);

	 Object columnNames[] = {"Distance(km)", "Observed anamolies (mGal)",
"Calculated anamolies (mGal)", "Depth(km)"};
		JTable table = new JTable(rowData, columnNames);
		table.setPreferredScrollableViewportSize(new Dimension(300,550));
		JScrollPane scrollPane = new JScrollPane(table);
		scrollPane.setAutoscrolls(true);
		POLYINV3D_MainPanel.p_East.add(scrollPane);
		POLYINV3D_MainPanel.p_East.validate();
		POLYINV3D_MainPanel.p_East.setVisible(true);

}

 public static void DrawTheContourPlot(String obs) {

	String		s;

	try {

		 s = obs;
			POLYINV3D_MainPanel.thePlot.getName("Observed anomaly");
			POLYINV3D_MainPanel.thePlot.setColor(Color.RED);
			POLYINV3D_MainPanel.thePlot.passObj(POLYINV3D_MainPanel.thePlot);

POLYINV3D_MainPanel.thePlot.ParseZedMatrix(s,POLYINV3D_AnomalyValues.cont);

POLYINV3D_MainPanel.thePlot.paint(POLYINV3D_MainPanel.thePlot.getGraphics(),POLYINV3D_A
nomalyValues.cont);
	}
	catch(ParseMatrixException e) {
			POLYINV3D_MainPanel.thePlot.repaint();

	}
	catch(IOException e) {
			POLYINV3D_MainPanel.thePlot.repaint();

	}
	finally {
```

```
                //System.out.println("Exiting DrawTheContourPlot");
        }
}
 public static void DrawTheContourPlot1(String obs) {

        String          s;

        try {
                s = obs;
                        POLYINV3D_MainPanel.thePlot1.getName("Modeled anomaly");
                        POLYINV3D_MainPanel.thePlot1.setColor(Color.MAGENTA);
                        POLYINV3D_MainPanel.thePlot1.passObj(POLYINV3D_MainPanel.thePlot1);

POLYINV3D_MainPanel.thePlot1.ParseZedMatrix(s,POLYINV3D_AnomalyValues.cont);

POLYINV3D_MainPanel.thePlot1.paint(POLYINV3D_MainPanel.thePlot1.getGraphics(),POLYINV3D
_AnomalyValues.cont);
        }
        catch(ParseMatrixException e) {
                        POLYINV3D_MainPanel.thePlot1.repaint();

        }
        catch(IOException e) {
                        POLYINV3D_MainPanel.thePlot1.repaint();

        }
        finally {
                //System.out.println("Exiting DrawTheContourPlot");
        }
}

 public static void DrawTheContourPlot2(String obs) {

        String          s;

        try {
                s = obs;
                        POLYINV3D_MainPanel.thePlot2.getName("Estimated Depth");
                        POLYINV3D_MainPanel.thePlot2.setColor(Color.blue.brighter());
                        POLYINV3D_MainPanel.thePlot2.passObj(POLYINV3D_MainPanel.thePlot2);

POLYINV3D_MainPanel.thePlot2.ParseZedMatrix(s,POLYINV3D_AnomalyValues.cont);

POLYINV3D_MainPanel.thePlot2.paint(POLYINV3D_MainPanel.thePlot2.getGraphics(),POLYINV3D
_AnomalyValues.cont);

POLYINV3D_MainPanel.thePlot2.index(POLYINV3D_MainPanel.thePlot2.getGraphics());
        }
        catch(ParseMatrixException e) {
                        POLYINV3D_MainPanel.thePlot2.repaint();

        }
        catch(IOException e) {
                        POLYINV3D_MainPanel.thePlot2.repaint();

        }
        finally {
                //System.out.println("Exiting DrawTheContourPlot");
        }
}
```

```
}
```

```
package com.polyinv3d.view;

import java.awt.*;
import java.io.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.awt.image.ImageObserver;
import javax.imageio.ImageIO;

import com.polyinv3d.model.POLYINV3D_AnomalyValues;



class ContourPlot extends Canvas {


 final static boolean          SHOW_NUMBERS     = true;
 final static int          BLANK              = 32,
                   OPEN_SUITE          = (int)'{',
                   CLOSE_SUITE          = (int)'}',
                   BETWEEN_ARGS          = (int)',',
                   N_CONTOURS=10,
                   PLOT_MARGIN          = 20,
                   WEE_BIT                  =  3,
                   NUMBER_LENGTH          =  3;
 final static double          Z_MAX_MAX       = 1.0E+10,
                   Z_MIN_MIN          = -Z_MAX_MAX;
 final static String EOL          =
                            System.getProperty("line.separator");


 int               xSteps, ySteps;
 float             z[][];
 boolean                    logInterpolation = false;
 Dimension          d;
 double                     deltaX, deltaY;
 String Name ="";
 Color col;
 ImageObserver io;


 int          l1[] = new int[4];
 int          l2[] = new int[4];
 int          ij[] = new int[2];
 int          i1[] = new int[2];
 int          i2[] = new int[2];
 int          i3[] = new int[6];
 int               ibkey,icur,jcur,ii,jj,elle,ix,iedge,iflag,ni,ks;
 int               cntrIndex,prevIndex;
 int               idir,nxidir,k;
 double          z1,z2,cval,zMax,zMin;
 double          intersect[] = new double[4];
```

```
double          xy[]      = new double[2];
double          prevXY[] = new double[2];

boolean          jump;


public ContourPlot(int x, int y) {
        super();
        xSteps = x;
        ySteps = y;

                setForeground(Color.black);
                setBackground(Color.white);
}

public void getName(String s){
        Name = s;
}

int          ncv = N_CONTOURS;
float          cv[];
public Graphics g1;

int sign(int a, int b) {
        a = Math.abs(a);
        if (b < 0)                  return -a;
        else                        return  a;
}


void InvalidData() {
        cv[0] = (float)0.0;
        cv[1] = (float)0.0;
}


        void GetExtremes() throws ParseMatrixException {
        int          i,j;
        double          here;

        zMin = z[0][0];
        zMax = zMin;
        for (j = 0; j < ySteps; j++) {
                for (i = 0; i < xSteps; i++) {
                                here = z[i][j];
                                if (zMin > here) zMin = here;
                                if (zMax < here) zMax = here;
                }
        }
        if (zMin == zMax) {
                InvalidData();
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errEqual);
        }
                return;
}


        void AssignContourValues() throws ParseMatrixException {
```

```
        int          i;
        double          delta;

        if ((logInterpolation) && (zMin <= 0.0)) {
                InvalidData();
                        throw new ParseMatrixException(POLYINV3D_MainPanel.errLog);
        }
        if (logInterpolation) {
                double          temp = Math.log(zMin);

                delta = (Math.log(zMax)-temp) / ncv;
                for (i = 0; i < ncv; i++) cv[i] = (float)Math.exp(temp + (i+1)*delta);
        }
        else {
                delta = (zMax-zMin) / ncv;
                for (i = 0; i < ncv; i++) cv[i] = (float)(zMin + (i+1)*delta);
        }
}


        String GetContourValuesString() {
        String          s = new String();
        int          i;

                for (i = 0; i < ncv; i++) s = s + "[" + Integer.toString(i) + "] " +
                        Float.toString(cv[i]) + EOL;
                return s;
}


        void SetMeasurements() {
        d = size();
        d.width  = 250;//d.width  - 2*PLOT_MARGIN;
        d.height = 250;//d.height - 2*PLOT_MARGIN;
        deltaY = d.width  / (ySteps - 1.0);
        deltaX = d.height / (xSteps - 1.0);
}

void setColor(Color s){

        col = s;
}

void DrawGrid(Graphics2D g) {
        int i,j,kx,ky;
        g.clearRect(0, 0, d.height+2*PLOT_MARGIN+10, d.width +2*PLOT_MARGIN);
        Color c = new Color(0.933f,0.914f,0.749f);
        g.setColor(c);
        g.fill(new Rectangle2D.Float( 50,20,250,250));
                g.setColor(Color.BLACK);
        for (i = 0; i < xSteps; i++) {
                if(i==0||i==xSteps-1)
                {
                        kx = (int)((float)i * deltaX);
                        g.drawLine( 30+PLOT_MARGIN,
                                        PLOT_MARGIN+kx,
                                        30+PLOT_MARGIN+d.height,
                                        PLOT_MARGIN+kx);
                }
        }
```

```
        g.drawLine( 30,290,300,290);
        g.drawLine( 30,290,30,20);
        float maxX = (float)
((POLYINV3D_AnomalyValues.input_mat_nx-1)*POLYINV3D_AnomalyValues.input_dx);
        float maxY = (float)
((POLYINV3D_AnomalyValues.input_mat_ny-1)*POLYINV3D_AnomalyValues.input_dy);
                int x = (int) POLYINV3D_AnomalyValues.input_dx;
                int y = (int) POLYINV3D_AnomalyValues.input_dy;
        int k=0;
        int l=0;
        while(x<=maxX){

                g.drawString("'",PLOT_MARGIN+30+250* x/maxX , 297);
                if(k%2==0)
                        g.drawString(""+x,PLOT_MARGIN+27+250* x/maxX, 310);

                        x =(int) (x+POLYINV3D_AnomalyValues.input_dx);
                k=k+1;
        }

        while(y<=maxY){

                g.drawString("-", 30,300-(250*y/maxY)-PLOT_MARGIN);
                if(l%2==0)
                                g.drawString(""+y,15,300-(250*y/maxY)-PLOT_MARGIN);

                        y =(int) (y+POLYINV3D_AnomalyValues.input_dy);
                l=l+1;
        }

        for (i = 0; i < xSteps; i++) {

                kx = (int)((float)i * deltaX);

        }

        for (j = 0; j < ySteps; j++) {
                if(j==0||j==ySteps-1)
                {
                        ky = (int)((float)j * deltaY);
                                g.drawLine(30+ PLOT_MARGIN+ky,
                                        PLOT_MARGIN,
                                                30+PLOT_MARGIN+ky,
                                        PLOT_MARGIN+d.width);
                }
        }


        g.setColor(col);
        g.setFont(new Font("Arial", 20, 20));
        g.drawString(""+Name,70,350);
        g.setFont(new Font("Arial", 20, 12));

}

 void SetColour(Graphics g) {
        Color c = new Color(
                                ((ncv-cntrIndex) * Color.blue.getRed()   +
                                        cntrIndex * Color.red.getRed())/ncv,
                                ((ncv-cntrIndex) * Color.blue.getGreen() +
```

```
                                        cntrIndex * Color.red.getGreen())/ncv,
                                ((ncv-cntrIndex) * Color.blue.getBlue()  +
                                        cntrIndex * Color.red.getBlue())/ncv);

        g.setColor(c);
}

        void DrawKernel(Graphics2D g2) {
        float           prevU,prevV,u,v;

        //DecimalFormat df =new DecimalFormat("0.#");
        if ((iflag == 1) || (iflag == 4) || (iflag == 5)) {
                if (cntrIndex != prevIndex) { // Must change colour

                        prevIndex = cntrIndex;
                }
                prevU =(float)((prevXY[0] - 1.0) * deltaX);
                prevV =(float) ((prevXY[1] - 1.0) * deltaY);
                u = (float)((xy[0] - 1.0) * deltaX);
                v = (float)((xy[1] - 1.0) * deltaY);

                SetColour(g2);
                if(cntrIndex==0){
                        g2.draw(new
Line2D.Float(30+PLOT_MARGIN+prevV,PLOT_MARGIN+prevU,30+PLOT_MARGIN+v,
PLOT_MARGIN+u));
                }
                g2.draw(new
Line2D.Float(30+PLOT_MARGIN+prevV,PLOT_MARGIN+prevU,30+PLOT_MARGIN+v,
PLOT_MARGIN+u));
                if ((SHOW_NUMBERS) && ((iflag==4) || (iflag==5))) {
                        g2.setColor(Color.black);
                    if      (u == 0)            u = u - WEE_BIT;
                    else if         (u == d.width)  u = u + PLOT_MARGIN/2;
                    else if         (v == 0)  v = v - PLOT_MARGIN/2;
                    else if          (v == d.height) v = v + WEE_BIT;
                    if(cntrIndex%3==0)

g2.drawString(""+POLYINV3D_AnomalyValues.df8.format(cv[cntrIndex]),
                                        PLOT_MARGIN+v+30, PLOT_MARGIN+u);

                }

        }
                prevXY[0] = xy[0];
                prevXY[1] = xy[1];
}

 void DetectBoundary() {
        ix = 1;
        if (ij[1-elle] != 1) {
                ii = ij[0] - i1[1-elle];
                jj = ij[1] - i1[elle];
                if (z[ii-1][jj-1] <= Z_MAX_MAX) {
                        ii = ij[0] + i2[elle];
                        jj = ij[1] + i2[1-elle];
                        if (z[ii-1][jj-1] < Z_MAX_MAX) ix = 0;
                }
                if (ij[1-elle] >= I1[1-elle]) {
                        ix = ix + 2;
```

```
                                        return;
                        }
                }
                ii = ij[0] + i1[1-elle];
                jj = ij[1] + i1[elle];
                if (z[ii-1][jj-1] > Z_MAX_MAX) {
                        ix = ix + 2;
                                return;
                }
                if (z[ij[0]][ij[1]] >= Z_MAX_MAX) ix = ix + 2;
        }

        boolean Routine_label_020() {
                l2[0] =  ij[0];
                l2[1] =  ij[1];
                l2[2] = -ij[0];
                l2[3] = -ij[1];
                idir = 0;
                nxidir = 1;
                k = 1;
                ij[0] = Math.abs(ij[0]);
                ij[1] = Math.abs(ij[1]);
                if (z[ij[0]-1][ij[1]-1] > Z_MAX_MAX) {
                        elle = idir % 2;
                        ij[elle] = sign(ij[elle],l1[k-1]);
                                return true;
                }
                elle = 0;
                        return false;
        }

        boolean Routine_label_050() {
                while (true) {
                        if (ij[elle] >= l1[elle]) {
                                if (++elle <= 1) continue;
                                elle = idir % 2;
                                ij[elle] = sign(ij[elle],l1[k-1]);
                                        if (Routine_label_150()) return true;
                                continue;
                        }
                        ii = ij[0] + i1[elle];
                        jj = ij[1] + i1[1-elle];
                        if (z[ii-1][jj-1] > Z_MAX_MAX) {
                                if (++elle <= 1) continue;
                                elle = idir % 2;
                                ij[elle] = sign(ij[elle],l1[k-1]);
                                        if (Routine_label_150()) return true;
                                continue;
                        }
                                break;
                }
                jump = false;
                        return false;
        }
        boolean Routine_label_150() {
                while (true) {
                        if (ij[elle] < l1[k-1]) {
                                ij[elle]++;
                                if (ij[elle] > l2[k-1]) {
                                        l2[k-1] = ij[elle];
```

```
                                idir = nxidir;
                                nxidir = idir + 1;
                                k = nxidir;
                                if (nxidir > 3) nxidir = 0;
                        }
                        ij[0] = Math.abs(ij[0]);
                        ij[1] = Math.abs(ij[1]);
                        if (z[ij[0]-1][ij[1]-1] > Z_MAX_MAX) {
                                elle = idir % 2;
                                ij[elle] = sign(ij[elle],l1[k-1]);
                                continue;
                        }
                        elle = 0;
                                return false;
                }
                if (idir != nxidir) {
                        nxidir++;
                        ij[elle] = l1[k-1];
                        k = nxidir;
                        elle = 1 - elle;
                        ij[elle] = l2[k-1];
                        if (nxidir > 3) nxidir = 0;
                        continue;
                }

                        if (ibkey != 0) return true;
                ibkey = 1;
                ij[0] = icur;
                ij[1] = jcur;
                if (Routine_label_020()) continue;
                        return false;
        }
}

short Routine_label_200(Graphics g,  boolean workSpace[])
{
        Graphics2D g2 = (Graphics2D) g;
        while (true) {
                xy[elle] = 1.0*ij[elle] + intersect[iedge-1];
                xy[1-elle] = 1.0*ij[1-elle];
                workSpace[2*(xSteps*(ySteps*cntrIndex+ij[1]-1)
                                +ij[0]-1) + elle] = true;
                        DrawKernel(g2);
                if (iflag >= 4) {
                        icur = ij[0];
                        jcur = ij[1];
                                return 1;
                }
                ContinueContour();
                if (!workSpace[2*(xSteps*(ySteps*cntrIndex
                                        +ij[1]-1)+ij[0]-1)+elle]) return 2;
                iflag = 5;                  // 5. Finish a closed contour
                iedge = ks + 2;
                if (iedge > 4) iedge = iedge - 4;
                intersect[iedge-1] = intersect[ks-1];
        }
}
        boolean CrossedByContour(boolean workSpace[]) {
         ii = ij[0] + i1[elle];
         jj = ij[1] + i1[1-elle];
```

```
        z1 = z[ij[0]-1][ij[1]-1];
        z2 = z[ii-1][jj-1];
        for (cntrIndex = 0; cntrIndex < ncv; cntrIndex++) {
                int i = 2*(xSteps*(ySteps*cntrIndex+ij[1]-1) + ij[0]-1) + elle;

                if (!workSpace[i]) {
                        float x = cv[cntrIndex];
                        if ((x>Math.min(z1,z2)) && (x<=Math.max(z1,z2))) {
                                workSpace[i] = true;
                                        return true;
                        }
                }
        }
                return false;
}
void ContinueContour() {
        short local_k;

        ni = 1;
        if (iedge >= 3) {
                ij[0] = ij[0] - i3[iedge-1];
                ij[1] = ij[1] - i3[iedge+1];
        }
        for (local_k = 1; local_k < 5; local_k++)
                if (local_k != iedge) {
                        ii = ij[0] + i3[local_k-1];
                        jj = ij[1] + i3[local_k];
                        z1 = z[ii-1][jj-1];
                        ii = ij[0] + i3[local_k];
                        jj = ij[1] + i3[local_k+1];
                        z2 = z[ii-1][jj-1];
                        if ((cval > Math.min(z1,z2) && (cval <= Math.max(z1,z2)))) {
                                if ((local_k == 1) || (local_k == 4)) {
                                        double          zz = z2;

                                        z2 = z1;
                                        z1 = zz;
                                }
                                intersect[local_k-1] = (cval - z1)/(z2 - z1);
                                ni++;
                                ks = local_k;
                        }
                }
        if (ni != 2) {
                ks = 5 - iedge;
                if (intersect[2] >= intersect[0]) {
                        ks = 3 - iedge;
                        if (ks <= 0) ks = ks + 4;
                }
        }
        elle = ks - 1;
        iflag = 1;                      // 1. Continue a contour
        jump = true;
        if (ks >= 3) {
                ij[0] = ij[0] + i3[ks-1];
                ij[1] = ij[1] + i3[ks+1];
                elle = ks - 3;
        }
}
        void ContourPlotKernel(Graphics2D g, boolean workSpace[])
```

```
{
        Graphics2D g2 = (Graphics2D) g;
        short val_label_200;

        l1[0] = xSteps;         l1[1] = ySteps;
        l1[2] = -1;l1[3] = -1;
        i1[0] =         1; i1[1] =  0;
        i2[0] =         1; i2[1] = -1;
        i3[0] =         1; i3[1] =  0; i3[2] = 0;
        i3[3] =         1; i3[4] =  1; i3[5] = 0;
                prevXY[0] = 0.0; prevXY[1] = 0.0;
        xy[0] = 1.0; xy[1] = 1.0;
        cntrIndex = 0;
                prevIndex = -1;
        iflag = 6;
                DrawKernel(g2);
        icur = Math.max(1, Math.min((int)Math.floor(xy[0]), xSteps));
        jcur = Math.max(1, Math.min((int)Math.floor(xy[1]), ySteps));
        ibkey = 0;
        ij[0] = icur;
        ij[1] = jcur;
        if (Routine_label_020() &&
                                Routine_label_150()) return;
                if (Routine_label_050()) return;
        while (true) {
                 DetectBoundary();
                 if (jump) {
                        if (ix != 0) iflag = 4; // Finish contour at boundary
                        iedge = ks + 2;
                        if (iedge > 4) iedge = iedge - 4;
                        intersect[iedge-1] = intersect[ks-1];
                        val_label_200 = Routine_label_200(g,workSpace);
                        if (val_label_200 == 1) {
                                        if (Routine_label_020() && Routine_label_150()) return;
                                        if (Routine_label_050()) return;
                                 continue;
                        }
                        if (val_label_200 == 2) continue;
                                return;
                }
                       if ((ix != 3) && (ix+ibkey != 0) && CrossedByContour(workSpace)) {
                        iedge = elle + 1;
                        cval = cv[cntrIndex];
                        if (ix != 1) iedge = iedge + 2;
                        iflag = 2 + ibkey;
                        intersect[iedge-1] = (cval - z1) / (z2 - z1);
                        val_label_200 = Routine_label_200(g,workSpace);
                        if (val_label_200 == 1) {
                                        if (Routine_label_020() && Routine_label_150()) return;
                                        if (Routine_label_050()) return;
                                 continue;
                        }
                        if (val_label_200 == 2) continue;
                                return;
                }
                if (++elle > 1) {
                        elle = idir % 2;
                        ij[elle] = sign(ij[elle],l1[k-1]);
                                if (Routine_label_150()) return;
                }
```

```
                        if (Routine_label_050()) return;
        }
}

public void passObj(ImageObserver s){

        io =s;
}
public void paint(Graphics g,int n)
{
        ncv=n;
        int workLength = 2 * xSteps * ySteps * ncv;
        boolean          workSpace[]; // Allocate below if data valid

                SetMeasurements();
        try {
                int width = 350;
                int height = 360;
                Buff          eredImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buff          er.createGraphics();

                        g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);

                Graphics2D g2 = (Graphics2D)g1 ;
                        g2.setBackground(Color.WHITE);
                DrawGrid(g2);
                        if (cv[0] != cv[1]) { // Valid data
                         workSpace = new boolean[workLength];
                                ContourPlotKernel(g2, workSpace);
                }

                        FileOutputStream os = new
FileOutputStream(POLYINV3D_AnomalyValues.input_area_name+Name+".jpg");
                ImageIO.write(buff          er, "jpg", os);
                os.close();
                        String path =  POLYINV3D_AnomalyValues.input_area_name+Name
+".jpg";
                Buff          eredImage image = ImageIO.read(new File(path));

                        Graphics g_image = g;//POLYINV3D_MainPanel.p_East.getGraphics();
                g_image.drawImage(image, -10, 100, image.getWidth(), image.getHeight(),io
);

        }
        catch (Exception e2) {

                        e2.printStackTrace();
        }

}
public void index(Graphics g){

        try {
                int width = 90;
                int height = 90;
                Buff          eredImage buffer = new
BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
                Graphics g1= buff          er.createGraphics();
```

```
                    g1.setColor(Color.WHITE);
                g1.fillRect(0,0,width,height);

                Graphics2D g2 = (Graphics2D)g1 ;
                    g2.setBackground(Color.WHITE);

                drawIndex(g2);


                    FileOutputStream os = new FileOutputStream("Index.jpg");
                ImageIO.write(buff          er, "jpg", os);
                os.close();
                String path =  "Index.jpg";
                Buff          eredImage image = ImageIO.read(new File(path));

                    Graphics g_image = g;//POLYINV3D_MainPanel.p_East.getGraphics();
                g_image.drawImage(image, 180, 20, image.getWidth(),
image.getHeight(),io );
        }
        catch (Exception e2) {

                    e2.printStackTrace();
        }



}

public void drawIndex(Graphics2D g){

                g.setColor(Color.black);

        g.drawLine(10, 70, 70, 70);
        g.drawLine(10, 70, 10,0);
        g.setFont(new Font("Arial", 40,11));
        g.drawString("EAST (km)", 20, 80);
        String a[]={"N","O","R","T","H","(k","m)"};
        for(int e=0;e<a.length;e++)
        {
                g.drawString(""+a[e],0,10+(e*10));
        }
}


public void ParseZedMatrix(String s,int n)
                    throws ParseMatrixException, IOException
{
        StringBuff          erInputStream i;
                StreamTokenizer          t;
        ncv=n;
        cv= new float[ncv+2];
        i = new StringBuff          erInputStream(s);
                t = new StreamTokenizer(i);

        z = null;  // Junk any existing matrix
        EatCharacter(t,OPEN_SUITE);
                do ParseRowVector(t);
                while (t.nextToken() == BETWEEN_ARGS);
        if (t.ttype != CLOSE_SUITE) {
```

```
                InvalidData();
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errExpect+(char)CLOSE_SUITE);
        }
                if (t.nextToken() != StreamTokenizer.TT_EOF) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errEOF);
        }
        MakeMatrixRectangular();
                GetExtremes();
        if (zMax > Z_MAX_MAX) zMax = Z_MAX_MAX;
        if (zMin < Z_MIN_MIN) zMin = Z_MIN_MIN;
                AssignContourValues();
}

        public void ParseRowVector(StreamTokenizer t)
                        throws ParseMatrixException, IOException
{
        if (z == null) z = new float[1][];
        else AddRow();
        EatCharacter(t,OPEN_SUITE);
        do {
                        if (t.nextToken() == StreamTokenizer.TT_NUMBER) {
                         int x = z.length - 1;

                         if (z[x] == null) {
                                 z[x] = new float[1];
                                 z[x][0] = (float)t.nval;
                         }
                         else AddColumn((float)t.nval);
                }
                else {
                         int x = z.length - 1;
                         int y = z[x].length - 1;

                         InvalidData();
                                 throw new ParseMatrixException(
                                                 POLYINV3D_MainPanel.errParse + EOL +
                                                 POLYINV3D_MainPanel.errComp + " [" +
                                                 Integer.toString(x) + "," +
                                                 Integer.toString(y) + "]");
                }
                } while (t.nextToken() == BETWEEN_ARGS);
        if (t.ttype != CLOSE_SUITE) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errExpect+(char)CLOSE_SUITE);
        }
}
        public void AddRow() throws ParseMatrixException {
         int leng = z.length;
         float temp[][];

                if (leng >= POLYINV3D_MainPanel.MAX_X_STEPS)
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
```

```
                                        POLYINV3D_MainPanel.errBounds);
        temp = new float[leng+1][];
        System.arraycopy(z, 0, temp, 0, leng);
        z = temp;
}
public void AddColumn(float val)
                        throws ParseMatrixException
{
        int i = z.length - 1;
        int leng = z[i].length;
        float temp[];

                if (leng >= POLYINV3D_MainPanel.MAX_Y_STEPS)
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errBounds);
        temp = new float[leng+1];
        System.arraycopy(z[i], 0, temp, 0, leng);
        temp[leng] = val;
        z[i] = temp;
}

public void MakeMatrixRectangular() {
        int             i,y,leng;

        xSteps = z.length;
                ySteps = POLYINV3D_MainPanel.MIN_Y_STEPS;
        for (i = 0; i < xSteps; i++) {
                y = z[i].length;
                if (ySteps < y) ySteps = y;
        }

        for (i = 0; i < xSteps; i++) {
                leng = z[i].length;
                if (leng < ySteps) {
                        float temp[] = new float[ySteps];

                        System.arraycopy(z[i], 0, temp, 0, leng);
                        while (leng < ySteps) temp[leng++] = 0;
                        z[i] = temp;
                }
        }
}
        public String ReturnZedMatrix() {
        String          s,oneValue;
        int             i,j;

        s = new String(
                                POLYINV3D_MainPanel.infoStrX + xSteps + EOL +
                                POLYINV3D_MainPanel.infoStrY + ySteps + EOL);
        for (i = 0; i < xSteps; i++) {
                for (j = 0; j < ySteps; j++) {
                                oneValue = Double.toString(z[i][j]);
                                while (oneValue.length() < NUMBER_LENGTH)
                                        oneValue = " " + oneValue;
                                s = s + oneValue;
                        if (j < ySteps-1) s = s + " ";
                }
                s = s + EOL;
        }
```

```
                return s;
}
        public void EatCharacter(StreamTokenizer t, int c)
                        throws ParseMatrixException, IOException
{
                while (t.nextToken() == BLANK) ;
         if (t.ttype != c) {
                 InvalidData();
                        throw new ParseMatrixException(
                                        POLYINV3D_MainPanel.errParse + EOL +
                                        POLYINV3D_MainPanel.errExpect + (char)c);
        }
}
}

class ContourPlotLayout extends java.lang.Object
{

        private static final int COUNT = POLYINV3D_MainPanel.NUMBER_COMPONENTS;
 private static final int
 MARGIN                  =   5,
 MIN_PLOT_DIMEN          = 300,
 LEFT_WIDTH          = 250,
 CBOX_WIDTH          = 130,
 BUTTON_H_POS            = MARGIN + CBOX_WIDTH + MARGIN,
 BUTTON_WIDTH            = LEFT_WIDTH - CBOX_WIDTH - MARGIN,
 LINE_HEIGHT          =  25,
        DATA_HEIGHT = 105,
 MIN_RES_HEIGHT          =  50,
        DATA_V_POS  = MARGIN + MARGIN + LINE_HEIGHT,
 BUTTON_V_POS                = DATA_V_POS + MARGIN + DATA_HEIGHT,
        RESULTS_V_POS       = BUTTON_V_POS + MARGIN + LINE_HEIGHT;
 Component k[]                   = new Component[COUNT];
 Dimension d             = new Dimension(MIN_PLOT_DIMEN, MIN_PLOT_DIMEN);
        int results_height        = MIN_RES_HEIGHT;

        public void GetDimensions(Container  parent) {
                d = parent.size();
         d.width = d.width - LEFT_WIDTH - 3*MARGIN;
         d.height = d.height - 2*MARGIN;
         if (d.width < MIN_PLOT_DIMEN) d.width = MIN_PLOT_DIMEN;
         if (d.height < MIN_PLOT_DIMEN) d.height = MIN_PLOT_DIMEN;
         if (d.width > d.height) d.width = d.height;
         else if (d.height > d.width) d.height = d.width;
                results_height = d.height + MARGIN - RESULTS_V_POS;
                if (results_height < MIN_RES_HEIGHT) results_height = MIN_RES_HEIGHT;
}
 public void addComponentNumber(int i, Component c) {
        if ((i < 0) || (i >= COUNT)) {
                        throw new ArrayIndexOutOfBoundsException();
        }
        else if (k[i] != null) {
                //                      throw new SomeKindOfException(
                //                              "Attempt to add a component already added");
        }
        else k[i] = c;
}

 public void layoutContainer(Container                  parent) {
                GetDimensions(parent);
```

```
            if (k[0] != null) k[0].reshape
        (2*MARGIN+LEFT_WIDTH,MARGIN,d.width,d.height);
            if (k[1] != null) k[1].reshape
        (MARGIN,MARGIN,LEFT_WIDTH,LINE_HEIGHT);
            if (k[2] != null) k[2].reshape
            (MARGIN,DATA_V_POS,LEFT_WIDTH,DATA_HEIGHT);
            if (k[3] !=null) k[3].reshape
        (MARGIN,BUTTON_V_POS,CBOX_WIDTH,LINE_HEIGHT);
            if (k[4] != null) k[4].reshape
        (BUTTON_H_POS,BUTTON_V_POS, BUTTON_WIDTH,LINE_HEIGHT);
            if (k[5] != null) k[5].reshape
            (MARGIN,RESULTS_V_POS,LEFT_WIDTH,results_height);
 }

        public Dimension minimumLayoutSize(Container  parent) {
            return new Dimension(
                    3*MARGIN + LEFT_WIDTH + MIN_PLOT_DIMEN,
                    2*MARGIN + MIN_PLOT_DIMEN);
 }

        public Dimension preferredLayoutSize(Container     parent) {
            GetDimensions(parent);
            return new Dimension(3*MARGIN + d.width + LEFT_WIDTH,
                    2*MARGIN + d.height);
 }

        public void removeLayoutComponent(Component c) {
            for (int i = 0; i < COUNT; i++) if (c == k[i]) k[i] = null;
 }
}

class ParseMatrixException extends Exception {
 public ParseMatrixException(String message) { super(message); }
}
```

```
package com.polyinv3d.model;

import java.text.DecimalFormat;
import java.util.HashMap;

import com.polyinv3d.util.POLYINV3D_Utility;
import com.polyinv3d.view.POLYINV3D_TableView;

public class POLYINV3D_AnomalyValues {
 public static Object obj[][] = null;
 public static double [][]gobse;
 public static double [][]gcalu;
 public static double [][]zdep;

 public static int input_iter = 0 ;
 public static int input_mat_nx = 0;
 public static int input_mat_ny = 0;
 public static double input_dx = 0;
 public static double input_dy = 0;
 public static int input_num_con_gr = 0;
 public static int input_num_con_dep = 0;
 double input_sd_poly=0;
```

```
double input_almda_st=0;

double input_zlt_km = 0;
final String EOL          =
                        System.getProperty("line.separator");

        public static String input_area_name ,input_pro_name= "";
public static String val=null;
public static String val1=null;
public static String val2=null;
public static int cont = 0;
public static DecimalFormat df8;

        public void getAnamolyValues(HashMap h_Map) {

         double input_nob_gob[] = null;

         try {

                        input_iter = POLYINV3D_Utility.convertInteger((String)h_Map.get("N_OBS"));
                input_nob_gob =
POLYINV3D_Utility.convertDoubleArray((String)h_Map.get("NOB_GOB"));
                input_sd_poly =
POLYINV3D_Utility.convertDouble((String)h_Map.get("SD_POLY"));
                input_almda_st =
POLYINV3D_Utility.convertDouble((String)h_Map.get("ALPHA_ST"));;
                input_mat_nx =
POLYINV3D_Utility.convertInteger((String)h_Map.get("MAT_NX"));
                input_mat_ny =
POLYINV3D_Utility.convertInteger((String)h_Map.get("MAT_NY"));
                        input_dx = POLYINV3D_Utility.convertDouble((String)h_Map.get("DIS_DX"));
                        input_dy = POLYINV3D_Utility.convertDouble((String)h_Map.get("DIS_DY"));
                input_zlt_km =
POLYINV3D_Utility.convertDouble((String)h_Map.get("OFF_ZLT"));
                input_num_con_gr =
POLYINV3D_Utility.convertInteger((String)h_Map.get("NUM_CON"));
                input_num_con_dep =
POLYINV3D_Utility.convertInteger((String)h_Map.get("NUM_CON1"));
                        input_area_name =
POLYINV3D_Utility.convertString((String)h_Map.get("AREA_FE"));
                        //input_pro_name =
POLYINV3D_Utility.convertString((String)h_Map.get("PROFILE_NAME"));
        }
        catch(Exception e) {
                        e.printStackTrace();
        }
        zdep = new double[input_mat_ny+2][input_mat_nx+2];
        gcalu = new double[input_mat_ny+2][input_mat_nx+2];
        gobse = new double[input_mat_ny+2][input_mat_nx+2];

        double conv[][] = new double[input_mat_ny+1][input_mat_nx+1];


        double x[] = new double[input_mat_nx+1];
        double y[] = new double[input_mat_ny+1];
        double g[]= new double[input_mat_ny*input_mat_nx+1];
        double gc[]= new double[input_mat_ny*input_mat_nx+1];
        double gc1[]= new double[input_mat_ny*input_mat_nx+5];
        double gc2[]= new double[input_mat_ny*input_mat_nx+5];
        double gs1[][] = new double[input_mat_ny+2][input_mat_nx+2];
```

```
        double err[]= new double[input_mat_ny*input_mat_nx+5];
        double par[]= new double[input_mat_ny*input_mat_nx+1];
        double par1[]= new double[input_mat_ny*input_mat_nx+5];
        double par2[]= new double[input_mat_ny*input_mat_nx+5];
        double zz[][] = new double[input_mat_ny+2][input_mat_nx+2];
        double zz1[][] = new double[input_mat_ny+2][input_mat_nx+2];
        double s[][]= new double[input_mat_ny*input_mat_nx+5]
[input_mat_ny*input_mat_nx+5];
        double p[][] = new double[input_mat_nx * input_mat_ny+1][input_mat_nx *
input_mat_ny+1];
        double []b = new double[input_mat_ny*input_mat_nx+5];
        double []ks = new double[2];
        double []dupar = new double[input_mat_ny*input_mat_nx+5];

        double a[]=new double[input_mat_nx*input_mat_ny+1];

        for (int i = 1;i <= input_mat_nx * input_mat_ny; i++) {

                a[i - 1] = input_nob_gob[i];

        }
        for (int I = 0;I < input_mat_ny; I++) {

                for (int K = 0;K < input_mat_nx; K++) {

                        conv[I][K] = a[(I * input_mat_nx) + K];

                }
        }

        for (int I = 1;I <= input_mat_ny; I++) {

                for (int K = 1;K <= input_mat_nx; K++) {

                        gobse[I][K]= conv[I-1][K-1];

                }

        }


        int nx= input_mat_nx;
        int ny= input_mat_ny;
        double          pi = 22.0 / 7.0;
        double          gk = 13.3333;
        double          aer = 0.001 * nx * ny;
        double          dpar = 0.05;
        int n = nx * ny;
        int         np = (nx - 2) * (ny - 2);
        int         np1 = np+1;
        double          lambda = 0.5;

        int JS=0;

        for(int I = 1;I <= ny; I++){
                for(int K = 1; K <= nx; K++){
                        JS=JS + 1;
                        g[JS] = gobse[I][K];
                }
        }
```

```
	for(int I = 1;I <= nx; I++){

		x[I] = (I - 1) * input_dx;

	}
	for(int I = 1;I <= ny; I++){

		y[I] = (I - 1) * input_dy;

	}
	for(int I = 2;I <= ny - 1; I++){

		for(int K = 2; K <= nx - 1; K++){

			zdep[I][K] = -(1 / input_almda_st) * Math.log(1 - ((input_almda_st *
gobse[I][K]) / (gk * pi * input_sd_poly)));
		}
	}


	JS=0;
	for(int I = 2;I <= ny - 1; I++){
		for(int K = 2; K <= nx - 1; K++) {
			JS=JS + 1;
			par[JS] = zdep[I][K];
		}
	}

		GANO(x, par, y, nx, ny, input_sd_poly, input_almda_st, gc);


	double funct1= 0.0;
	double funct2= 0.0;
	for(int I = 1; I <= ny * nx; I++){

		err[I] = g[I] - gc[I];
		funct1 = funct1 + Math.pow(err[I], 2);

	}
	System.out.println(funct1);
	for(int ITER=1;ITER<=input_iter+1;ITER++) {
		System.out.println(ITER);

		JS=0;
		for(int I = 2; I <= ny - 1; I++){
			for(int K = 2; K <= nx - 1; K++){

				JS = JS + 1;
				zz[I][K] = par[JS];

			}
		}

		JS=0;
		for(int I = 1;I <= ny; I++){
			for(int K = 1;K <= nx; K++){
				JS = JS + 1;
				gs1[I][K] = gc[JS];
				gcalu[I][K] = gc[JS];
```

```
		}
	}

	JS=0;
	for(int I = 2;I <= ny - 1; I++){
		for(int K = 2; K <= nx - 1; K++){
			JS = JS + 1;
			zz1[I][K] = par[JS];
			zdep[I][K] = par[JS];
		}
	}


	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;
			GANO(x, par1, y, nx, ny, input_sd_poly, input_almda_st, gc1);
		par1[I] = par[I] - dpar / 2.0;
			GANO(x, par1, y, nx, ny, input_sd_poly, input_almda_st, gc2);
		for(int K = 1; K <= n; K++){

			s[I][K] = (gc1[K] - gc2[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 <= n; 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 <= n; 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;

                        }
                }
        }

        SIMEQ(p,b,np,ks);
        for(int I= 1; I <= np; I++){
                par2[I] = dupar[I] + b[I];
                if(par2[I] < 0) par2[I] = 0;
                if(par2[I] > input_zlt_km) par2[I] = input_zlt_km;}
               GANO(x, par2, y, nx, ny, input_sd_poly, input_almda_st, gc);
        funct2 = 0.0;
        for(int K = 1; K <= n; K++){

                err[K] = g[K] - gc[K];
                funct2 = funct2 + Math.pow(err[K], 2);

        }

        if (funct1 - funct2 < 0) {

                lambda = lambda * 2;

                if (lambda - 13 < 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);

for(int I = 1; I <= np; I++){

        par[I] = par2[I];

}
lambda = lambda / 2.0;


{
        String a1 = "{";
        String a21 = "{";
        String a31 = "{";
        String a2 = "}";
        String a4 = "{";
        String a3[] = new String[input_mat_nx+1];

        for(int k = 0; k < input_mat_nx; k++) {
```

```
                    a3[k]="{";
                }

                    double regob[][] = new double[input_mat_ny*5][input_mat_nx*5];
                    double regcal[][] = new double[input_mat_ny+1][input_mat_nx+1];
                    double rezcal[][] = new double[input_mat_ny+1][input_mat_nx+1];
                String b1[][] = new String[input_mat_ny*5][input_mat_nx*5];
                String b2[][] = new String[input_mat_ny+1][input_mat_nx+1];
                String b3[][] = new String[input_mat_ny+1][input_mat_nx+1];

                for(int I = 0; I <= input_mat_ny; I++) {



                    for(int K = 0;K <= input_mat_nx; K++) {


                        DecimalFormat df =new DecimalFormat("0.####");


                            regob[I][K] = gobse[ I][K];
                            regcal[I][K] = gs1[ I][K];
                            rezcal[I][K] = zz1[ I][K];
                            b1[I][K] = Double.toString(regob[I][K]);
                            b2[I][K] = df.format(regcal[I][K]);//
Double.toString(regcal[K][I]);
                            b3[I][K] = df.format(rezcal[I][K]);//
Double.toString(rezcal[K][I]);

                        if(K < (input_mat_nx )) {

                            b1[I][K] = b1[I][K] + ",";
                            b2[I][K] = b2[I][K] + ",";
                            b3[I][K] = b3[I][K] + ",";
                        }
                        a1 = a1 + b1[I][K];
                        a21 = a21 + b2[I][K];
                        a31 = a31 + b3[I][K];
                    }

                    if (I < (input_mat_ny )) {

                        a1 = a1 + a2 + "," + EOL + a3[I];
                        a21 = a21 + a2 + "," + EOL + a3[I];
                        a31 = a31 + a2 + "," + EOL + a3[I];
                    }
                }
                a1 = a4 + a1 + a2 + a2;
                a21 = a4 + a21 + a2 + a2;
                a31 = a4 + a31 + a2 + a2;
                val = a1;
                val1 = a21;
                val2 = a31;
                a1 = null;
                a21 = null;
                a31 = null;
            }

            cont = input_num_con_gr;
```

```
                df8 = new DecimalFormat("0");

POLYINV3D_TableView.DrawTheContourPlot(POLYINV3D_AnomalyValues.val);

POLYINV3D_TableView.DrawTheContourPlot1(POLYINV3D_AnomalyValues.val1);
                cont = input_num_con_dep;
                df8 = new DecimalFormat("0.#");

POLYINV3D_TableView.DrawTheContourPlot2(POLYINV3D_AnomalyValues.val2);



                        setGraphValues(input_iter,gobse,gs1,zz1,input_mat_ny,input_mat_nx);



                System.out.println(funct2);

                if (funct2 - funct1 <= 0 ) {
                        if (funct2 - aer <= 0) {
                                        break;
                        }
                        else  {
                                funct1 = funct2;
                        }
                }
                else
                                break;



        }


}



        public double[] GANO(double []x,double []par,double []y,int nx,int ny,double sd,double
la,double []gc){


         double [][]zv = new double[ny + 2][nx + 2];
         double []yy = new double[ny + 2];
         double []xx = new double[nx + 2];
         double []effy = new double[3];
         double []zt = new double[3000];
         double []x1 = new double[300];
         double []gs = new double[10000];
         double []gc1 = new double[300];
         double ggc = 0,dz = 0,dc = 0,y2=0;
         double []gg2 = new double[3];

         int JS=0;
         for(int I = 2; I<= ny - 1; I++){
                for(int K = 2; K <= nx - 1; K++){
                        JS = JS + 1;
                        zv[I][K] = par[JS];
                        continue;
```

```
        }
}

for (int k = 1;k <= input_mat_nx; k++) {
        zv[1][k] = 0;
        zv[input_mat_ny][k] = 0;
}
for (int k = 2;k <= input_mat_ny; k++) {
        zv[k][1] = 0;
        zv[k][input_mat_nx] = 0;
}


int iit = 0;
for(int K11 = 1; K11 <= ny; K11++){
        for(int I11 = 1; I11 <= nx; I11++){
                iit = iit + 1;
                for(int IY = 1; IY <= ny; IY++){

                        yy[IY] = y[IY] - y[K11];

                }
                for(int IX=1;IX<=nx;IX++){

                        xx[IX]=x[IX]-x[I11];

                }
                Ival:for(int I = 1; I <= ny; I++){

                        effy[1] = 0.5 - yy[I];
                        effy[2] = 0.5 + yy[I];
                        double dg = 0.0;
                        xx[nx + 1]=xx[1];
                        zv[I][nx + 1]=zv[I][1];
                        Kval:for(int K = 1;K <= nx;K++){
                                //            int K1=K+1;
                                double dzz = zv[I][K + 1] - zv[I][K];
                                double  dx = xx[K + 1] - xx[K];
                                double r = Math.sqrt(dx * dx + dzz * dzz);

                                //if(R==0.0)//GO TO 9
                                        //break ;
                                double c  = dx / r;
                                double s = dzz / r;
                                double ct = c / s;
                                double dx1 = (xx[2] - xx[1]) / 4;
                                double zb = Math.abs(zv[I][K + 1] - zv[I][K]);
                                int nd = (int) (zb / dx1) + 1;
                                int n1 = nd / 2;
                                if (nd - (2 * n1 ) < 0 || nd - ( 2 * n1 ) > 0) {
                                        nd = nd + 1;
                                }
                                dz = zb / nd;
                                int n2 = nd + 1;




                                if(zv[I][K + 1] - zv[I][K] == 0)
                                        //zv[I][K]=zv[I][K]+0.0001;
```

```
                                        continue Kval ;

                                    Jzval:for(int jz = 1; jz <= n2; jz++){



                                        if(zv[l][K + 1] - zv[l][K] > 0)                {
                                            zt[jz] = zv[l][K] + dz * (jz-1);
                                            if(zt[jz] <= 0)
                                                zt[jz] = 0.001;
                                            x1[jz] = xx[K] + (zt[jz] - zv[l][K]) * ct;
                                            if(Math.abs(x1[jz]) < 0.01)
                                                x1[jz] = 0;
                                            //continue Kval;
                                        }
                                        if(zv[l][K + 1] - zv[l][K] < 0) {
                                            zt[jz] = zv[l][K] - dz * (jz-1);
                                            if(zt[jz] <= 0)
                                                zt[jz] = 0.001;
                                            x1[jz] = xx[K] + (zt[jz] - zv[l][K]) * ct;
                                            if(Math.abs(x1[jz]) < 0.01)
                                                x1[jz] = 0;
                                            //continue;
                                        }




                                    }


                                    for(int JZ = 1; JZ <= n2; JZ++){
                                        dc = sd * Math.exp(-la * zt[JZ]);
                                        for(int LK = 1; LK <= 2; LK++){
                                            y2 = effy[LK];
                                            double a = x1[JZ] - zt[JZ] * ct;
                                            double anum = y2 * (a + zt[JZ] * ct);
                                            double den1 = Math.sqrt(Math.pow((a
+ zt[JZ] * ct),2) + Math.pow(y2, 2) + Math.pow(zt[JZ], 2));
                                            double den = zt[JZ] * den1;
                                            gg2[LK] = -13.3333 * dc *
Math.atan(anum / den);
                                        }
                                        gs[JZ] = (gg2[1] + gg2[2]) / 2.0;
                                    }

                                    ggc=SIMP(gs,zt,n2,ggc);
                                    dg=dg+ggc;

                                }
                                gc1[l]=dg;


                            }
                            double gr=0.0;

                            for(int KK = 1; KK <= ny - 1; KK++){
                                double G1 = gc1[KK];
                                double G2 = gc1[KK + 1];
                                double DY = yy[KK + 1]-yy[KK];
```

```
                                gr = gr + DY * EXPNTR(G1, G2);

                        }
                        gc[iit]=gr;

                }
        }


                return(gc);


}

public static double  EXPNTR(double g1, double g2){

        double c;
        double expnt;
        if (g1 == 0)
                g1 = 0.0001;
        if(g2 == 0)
                g2 = 0.0001;
        if( g2 / g1 < 0) {
                double x = Math.abs(g1) / (Math.abs(g1) + Math.abs(g2));
                double  g = 0.00001 * g1 / Math.abs(g1);
                c = Math.log(g / g1);
                double f1 = ( g - g1) / c;
                g = 0.00001 * g2 / Math.abs(g2);
                c = Math.log(g2 / g);
                double f2 = (g2 - g) / c;
                expnt = f1 * x + f2 * (1 - x);
        }
        else {
                c = Math.log(g2 / g1);


                if(Math.abs(c) <= 0.0001){
                        expnt = g1;
                }
                else {
                        expnt = (g2 - g1) / c;
                }
                //expnt = dx * expnt;
        }
               return expnt;

}


public static double SIMP(double []gs,double []z,int n, double ggc) {

        ggc=0;
        double dz = z[2] - z[1];
        double sum1 = 0;
        double sum2 = 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;
		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,double [][]GOBS,double [][]GCAL,double [
][]dep,int ny,int nx) {

        obj = new Object[nx*ny*5][4];
        int K1;
        DecimalFormat df =new DecimalFormat("0.###");
        for (int i=0; i<= ny;i++) {

                int col = i*nx;
                if(i<ny)
                        obj[col+i][0] = "PROFILE :"+(i+1);
                for (int K = 0;K <= nx;K++){
                        if(K<nx&& i<ny){
                                K1=K+1;
                                obj[col+K+i+1][0] = "" + K1;
                                obj[col+K+i+1][1] = "" + df.format(GOBS[i+1][K+1]);
```

```
                                obj[col+K+i+1][2] = "" + df.format(GCAL[i+1][K+1]);
                                obj[col+K+i+1][3] = "" + df.format(dep[i+1][K+1]);
                        }
                }
        }
}



}
```

```
package com.polyinv3d.control;

import java.awt.event.*;
import java.io.File;
import java.text.DecimalFormat;

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

import com.polyinv3d.model.POLYINV3D_AnomalyValues;
//import com.polyinv3d.model.POLYINV3D_CalculateValues;
import com.polyinv3d.view.POLYINV3D_MainPanel;
import com.polyinv3d.view.POLYINV3D_TableView;



public class POLYINV3D_Controller implements ActionListener{



        com.polyinv3d.model.POLYINV3D_AnomalyValues cv = new
com.polyinv3d.model.POLYINV3D_AnomalyValues();
        Object rowdata[][]={};
 public static boolean success=false;
 public static boolean success1,success2,success3=false;

 public void actionPerformed(ActionEvent ae) {

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


                        com.polyinv3d.view.POLYINV3D_TableView.populateEastPanel(rowdata);

cv.getAnamolyValues(com.polyinv3d.view.POLYINV3D_MainPanel.captureValues());

com.polyinv3d.view.POLYINV3D_TableView.populateEastPanel(POLYINV3D_AnomalyValues.obj);
                        POLYINV3D_MainPanel.p_Center.addMouseListener(new MouseAction());
                        POLYINV3D_AnomalyValues.obj = null;
                        com.polyinv3d.view.POLYINV3D_MainPanel.p_East.repaint();
                        com.polyinv3d.view.POLYINV3D_MainView mv = new
com.polyinv3d.view.POLYINV3D_MainView();
                        mv.setResizable(true);

        }          else if(ae.getActionCommand().equals("Contour")){
```

```
                        POLYINV3D_AnomalyValues.cont =
POLYINV3D_AnomalyValues.input_num_con_gr;
                        POLYINV3D_AnomalyValues.df8 =new DecimalFormat("0");

POLYINV3D_TableView.DrawTheContourPlot(POLYINV3D_AnomalyValues.val);

POLYINV3D_TableView.DrawTheContourPlot1(POLYINV3D_AnomalyValues.val1);
                        POLYINV3D_AnomalyValues.cont =
POLYINV3D_AnomalyValues.input_num_con_dep;
                        POLYINV3D_AnomalyValues.df8 =new DecimalFormat("0.#");

POLYINV3D_TableView.DrawTheContourPlot2(POLYINV3D_AnomalyValues.val2);

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

                try
                {
                                POLYINV3D_PrintValues.printGraphValues();

                }
                catch(Exception e1) {

                                e1.printStackTrace();
                }

        }else if(ae.getActionCommand().equals("Load data")){
                        POLYINV3D_MainPanel.loadData();
                        //POLYINV3D_MainPanel.setDefaultValues();
        }else if(ae.getActionCommand().equals("Clear")){


                        POLYINV3D_MainPanel.clearDefaultValues();
                        POLYINV3D_MainPanel.clearPanel(POLYINV3D_MainPanel.p_Center);
                        com.polyinv3d.view.POLYINV3D_TableView.populateEastPanel(rowdata);
        }
        else if(ae.getActionCommand().equals("Exit")){
                JFrame frame = null;

                int r =         JOptionPane.showConfirmDialog(
                                  frame,
                                          "Exit POLYINV3D ?",
                                  "Confirm Exit ",
                                  JOptionPane.YES_NO_OPTION);
                if(r == JOptionPane.YES_OPTION ){
                                 if(POLYINV3D_Controller.success==false){
                                  String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Observed anomaly.jpg";
                                  File f = new File(fileName);
                                  f.delete();
                        }
                                 if(POLYINV3D_Controller.success1==false){
                                  String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Modeled anomaly.jpg";
                                  File f = new File(fileName);
                                  f.delete();
                        }
                                 if(POLYINV3D_Controller.success2==false){
                                  String fileName =
POLYINV3D_AnomalyValues.input_area_name+"Estimated Depth.jpg";
                                  File f = new File(fileName);
```

```
                                f.delete();
                        }
                                if(POLYINV3D_Controller.success3==false){
                                String fileName = "Index.jpg";
                                File f = new File(fileName);
                                f.delete();
                        }
                        System.exit(0);
                }
        }
 }

}

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

                POLYINV3D_TableView.DrawTheContourPlot(POLYINV3D_AnomalyValues.val);
                POLYINV3D_TableView.DrawTheContourPlot1(POLYINV3D_AnomalyValues.val1);
                POLYINV3D_TableView.DrawTheContourPlot2(POLYINV3D_AnomalyValues.val2);
 }

}
```

```
package com.polyinv3d.control;

import java.io.File;
import java.io.FileWriter;
import java.text.DecimalFormat;
import javax.swing.JFileChooser;

import com.polyinv3d.model.POLYINV3D_AnomalyValues;


public class POLYINV3D_PrintValues {


        public static void printGraphValues() throws Exception {
         try{
                        String current = System.getProperty("user.dir");
                        File img_file = new File(POLYINV3D_AnomalyValues.input_area_name
+"Observed anomaly.jpg");
                        File img_file1 = new File(POLYINV3D_AnomalyValues.input_area_name
+"Modeled anomaly.jpg");
                        File img_file2 = new File(POLYINV3D_AnomalyValues.input_area_name
+"Estimated Depth.jpg");
                File img_file3 = new File("Index.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());
```

```
                                        POLYINV3D_Controller.success = img_file.renameTo(new
File(dir,img_file.getName()));
                                        POLYINV3D_Controller.success1 = img_file1.renameTo(new
File(dir,img_file1.getName()));
                                        POLYINV3D_Controller.success2 = img_file2.renameTo(new
File(dir,img_file2.getName()));
                                        POLYINV3D_Controller.success3 = img_file3.renameTo(new
File(dir,img_file3.getName()));
                                        myWriter = new FileWriter(OutFile+".html");
                                        myWriter.write(" </table> </td> <td> <img src =
'"+"Index.jpg'");
                                        myWriter.write(" </table> </td> <td> <img src = '"+
POLYINV3D_AnomalyValues.input_area_name +"Observed anomaly.jpg'>"+"<img src = '"+
POLYINV3D_AnomalyValues.input_area_name +"Modeled anomaly.jpg'>"+"<img src = '"+
POLYINV3D_AnomalyValues.input_area_name +"Estimated Depth.jpg'</td></tr></table>");
                                        myWriter.write("<html> <Body onLoad = \"window.print()
\"><table> <tr> <td>" +
                                                        "<table border = 1> <tr> <th colspan =
4>LOCATION:- "+POLYINV3D_AnomalyValues.input_area_name+"</th> </tr>");


                                DecimalFormat df =new DecimalFormat("0.###");
                                        myWriter.write(" <tr><th colspan = 4> ITERATION"+"
"+POLYINV3D_AnomalyValues.input_iter+" </th></tr>");

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

                                        for(int i = 1;i <= POLYINV3D_AnomalyValues.input_mat_ny; i+
+) {

                                                myWriter.write("<tr><td>PROFILE NO :" + i+"</td></
tr>");
                                         for ( int K = 1; K <=
POLYINV3D_AnomalyValues.input_mat_nx; K++){

                                                        myWriter.write("<tr> <td>" + K+"</td>
<td>"+df.format(POLYINV3D_AnomalyValues.gobse[i][K])+"</td>
<td>"+df.format(POLYINV3D_AnomalyValues.gcalu[i][K])+"</td>
<td>"+df.format(POLYINV3D_AnomalyValues.zdep[i][K])+"</td></tr>");
                                         }

                                }


                                        myWriter.close();
                        }
                }
                else
                {
                                //pops up error message
                }
        }
        catch(Exception e1) {

                        e1.printStackTrace();
        }

 }
```

```
}
```

```
package com.polyinv3d.util;

public class POLYINV3D_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 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;
}
}
```

References

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.

Al-Garni, M.A., 2008, Walsh Transforms for depth determination of a finite vertical cylinder from its residual gravity anomaly: Symposium on the Application of Geophysics to Engineering and Environmental Problems Proceedings, 689-702. DOI:10.4133/1.2963311.

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.

Ashena, B.Z., and Ardestani, E.V., 2012, Depth estimation of gravity anomalies using modified Hilbert transform: SEG Global Meeting Abstracts, 1-4.

Athy, L.F., 1930, Density, porosity and compaction of sedimentary rocks: Bulletin of the American Association of Petroleum Geology, 14, 1-24.

Avseth, P., Mukerji, T., Mavko, G., and Tyssekvam, J. A., 2001, Rock physics and AVO analysis for lithofacies and pore fluid prediction in a North Sea oil field: The Leading Edge, 20, 429–434.

Ayala, C., Rubio, F.M., Rey-Moral, C., Reguera, M.I., and Biete, C., 2019, 3D Geophysical characterization of the La Rambla and Zafra de Záncara anticlines (Loranca basin, Central Spain): Geophysical Prospecting, 67, 580-594.

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.

Bal, O. T., and Kara, I., 2012, 3-D Gravity modeling of basins with vertical prisms: Application to Salt Lake region (Turkey): Journal of the Balkan Geophysical Society, 15 (1), 1-6.

Barbosa, V. C. F., and Silva, J.B.C., 2011, Reconstruction of geologic bodies in depth associated with a sedimentary basin using gravity and magnetic data: Geophysical Prospecting, 59(6), 1021–1034.

Barbosa, V.C.F., 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.F., Silva, J.B.C., Medeiros, W., 1999, Gravity inversion of a discontinuous relief stabilized by weighted smoothness constraints on depth: Geophysics, 64, 1429-1437.

Beiki, M., 2010, Analytic signals of gravity gradient tensor and their application to estimate source location: Geophysics, 75, I59–I74.

Banerjee, B., and Das Gupta, S.P., 1977, Short note: Gravitational attraction of a rectangular parallelepiped: Geophysics, 42, 1053–1055.

Bezada, M.J., and Zelt, C.A., 2011, Gravity inversion using seismically derived crustal density models and genetic algorithms: an application to the Caribbean-South American Plate boundary: Geophysical Journal International, 185(2), 577–592.

Bhattacharyya, B.K., and Leu, L.K., 1975, Spectral analysis of gravity and magnetic anomalies due to two-dimensional structures: Geophysics, 40(6), 993-1013.

Bhimasankaram, V.L.S., Mohan, N.L., Seshagiri Rao, S.V., 2006, Analysis of the gravity effect of two dimensional trapezoidal prisms using Fourier transform: Geophysical Prospecting, 25(2), 334-341.

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.

Biswas, A., 2016, Interpretation of gravity and magnetic anomaly over thin sheet-type structure using very fast simulated annealing global optimization technique: Modeling Earth Systems and Environment, 2, Article number: 30.

Biswas, A., 2015, Interpretation of residual gravity anomaly caused by simple shaped bodies using very fast simulated annealing global optimization: Geoscience Frontiers, 696), 875-893.

Biswas, A., Parija, M.P., Sushil Kumar, 2017, Global nonlinear optimization for the interpretation of source parameters from total gradient of gravity and magnetic anomalies caused by thin dyke: Annals of Geophysics, 60(2), G0218. DOI: 10.4401/ag-7129.

Blakely, R.J., 1995, Potential Theory in Gravity and Magnetic Applications: Cambridge University Press, 464.

Bond, J., 1996, Tectono-sedimentary evolution of the Almazán Basin, NE Spain. In: Friend, F., Dabro, C.J. (Eds.), Tertiary Basins of Spain: The Stratigraphic Record of Crustal Kinematics. Cambridge University Press, Cambridge, 203-213.

Boschetti, F., Dentith, M., List, R., 1997, Inversion of potential field data by genetic algorithms: Geophysical Prospecting, 45, 461-478.

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.

Bovanloo, R.H., Goudarzi, A., Ardestani, V.E.Z., and Riahi, M.A., 2012, Automatic gravity data quality improvement using undecimated discrete wavelet transform technique: SEG Global Meeting Abstracts, 1-4, https://doi.org/10.1190/IST092012-001.77

Cai, H., Xiong, B., and Zhu, Y., 2018, 3D Modeling and Inversion of Gravity Data in Exploration Scale. Intech Open Limited. https://www.intechopen.com/books/gravity-geoscience-applications-industrial-technology-and-quantum-aspect/3d-modeling-and-inversion-of-gravity-data-in-exploration-scale.

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.

Chakravarthi, V., 2020, Geodesy, Physical. In: Gupta H.K. (eds) Encyclopedia of Solid Earth Geophysics. Encyclopedia of Earth Sciences Series. Springer, Cham., https://doi.org/ 10. 1007/978-3-030-10475-7_227-1

Chakravarthi, V., Mallesh, K., Ramamma, B., 2017a, Basement depths estimation from gravity anomalies: Two 2.5D approaches coupled with exponential density contrast model, Journal of Geophysics and Engineering, 20, 303-315.

Chakravarthi, V., Pramod Kumar, M., Ramamma, B., Rajeswara Sastry, S., 2017b, Gravity anomaly interpretation of 2D fault morphologies by means of non-planar fault planes and exponential density contrast model: A space domain technique, Arabian Journal of Geosciences, doi:10.1007/s12517-017-2845-z.

Chakravarthi, V., Pramod Kumar, M., Ramamma, B., Rajeswara Sastry, S., 2016, 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, 124, 54-61.

Chakravarthi, V., Pramod Kumar, M., 2015a, 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., Ramamma, B., 2015b, Determination of Sedimentary Basin Basement Depth: A Space Domain Based Gravity Inversion using Exponential Density Function: Acta Geophysica, 63 (4), 1066-1079.

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., Venkat Reddy, T., 2013b, 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, Harsh K. Gupta (Ed.), 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., 2009, Automatic gravity inversion for simultaneous estimation of model parameters and regional gravity background: an application to 2D pull-apart basins: Current Science, 96 (10), 1349-1360.

Chakravarthi, V., Sundararajan, N., 2007a, INV2P5DSB - A code for gravity inversion of 2.5-d sedimentary basins using depth dependent density, Computers & Geosciences, 33, 449-456.

Chakravarthi, V., Sundararajan, N., 2007b, 3D gravity inversion of basement relief – A depth dependent density approach: Geophysics, 72, I23- I32.

Chakravarthi.V., Shankar,G. B. K., Muralidharan, D., Harinarayana, T., and Sundararajan, N., 2007c, An integrated geophysical approach for imaging subbasalt sedimentary basins: Case study of Jam River basin, India: Geophysics, 72, B141–B147**.**

Chakravarthi, V., Sundararajan, N., 2006a, 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., and Sundararajan, N., 2006b, Gravity anomalies of multiple prismatic structures with varying density - A Marquardt inversion: Pure and Applied Geophysics, 163, 229-242.

Chakravarthi, V., and Sundararajan, N., 2004a, Automatic 3-D gravity modeling of sedimentary basins with density contrast varying parabolically with depth. Computers & Geosciences, 30, 601-607.

Chakravarthi, V., and Sundararajan, N., 2004b, 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., Raghuram. H.M., and Singh, S.B., 2002, 3D Forward gravity modeling of density interfaces above which the density contrast varies continuously with depth: Computers & Geosciences, 28, 53-57.

Chapin, D.A., 1997, Wavelet transforms: A new paradigm for interpreting gravity and magnetics data? SEG Technical Program Expanded Abstracts, 486-489.

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.

Chen, C., Chen, Y., and Bian, S., 2019a, Evaluation of the spherical harmonic coefficients for the external potential of a polyhedral body with linearly varying density: Celestial Mechanics and Dynamical Astronomy, 131(2), DOI: 10.1007/s10569-019-9885-5.

Chen, C., Bian, S., and Li, H.A., 2019b, A spectral-domain approach for gravity forward modelling of 2D bodies: Journal of Geodesy, 93, 2123-2144.

Chen, C., Ouyang, Y., Bian, S., 2019, Spherical Harmonic Expansions for the Gravitational Field of a Polyhedral Body with Polynomial Density Contrast: Surveys in Geophysics, 40,197–246.

Chen, C., Ren, Z., Pan, K., Tang, J., Kalscheuer, T., Maurer, H., Sun, Y., and Li, Y., 2018, Exact solutions of the vertical gravitational anomaly for a polyhedral prism with vertical polynomial density contrast of arbitrary orders: Geophysics Journal International, 214, 2115–2132.

Collins, S.J., Dodds, A.R., Johnson, B.D., 1974, Gravity profile interpretation using Fourier transform: Geophysics, 39, 862-866.

Comacho, A. G., Montesinos, F.G., and Vieira, R., 1997, A three-dimensional gravity inversion applied to Sao Miguel Island Azores: Journal of Geophysical Research B, Solid Earth and Planets, 102, 7717–7730.

Corbato, C.E., 1965, A least-squares procedure for gravity interpretation: Geophysics, 30, 228-233.

Cordell, L., 1973, Gravity anomalies using an exponential density-depth function-San Jacinto Graben, California: Geophysics, 38, 684-690.

Cordell, L., and Henderson, R.G., 1968, Iterative three-dimensional solution of gravity anomaly data using a digital computer: Geophysics, 33 (4), 596-601.

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.

Dai, S., Zhao, D., Wang, S., Xiong, B., Zhang, Q., Li, K., Chen, L., and Chen, Q., 2019, Three-dimensional numerical modeling of gravity and magnetic anomaly in a mixed space-wavenumber domain, Geophysics, 84(4), G41-G54.

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.

Debeglia, N., and Corpell, J., 1997, Automatic 3-D interpretation of potential field data using analytic signal derivatives: Geophysics, 62, 87–96.

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(1), 27–38.

Dogru, F.N, and Pamukcu, O., 2016, Investigation of boundaries of discontinuities using wavelet transform: Eastern part of Turkey: SEG Technical Program Expanded Abstracts, 1612-1616.

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, eds., 65–75.

D'Urso, M. G., and Trotta, S., 2017, Gravity Anomaly of Polyhedral Bodies Having a Polynomial Density Contrast: Surveys in Geophysics, 38 (4), 781–832.

D'Urso M.G., 2015a, A Remark on the Computation of the Gravitational Potential of Masses with Linearly Varying Density. In: Sneeuw N., Novák P., Crespi M., Sansò F. (eds) VIII Hotine-Marussi Symposium on Mathematical Geodesy. International Association of Geodesy Symposia, vol 142. Springer, Cham, DOI: https://doi.org/10.1007/ 1345_ 2015_138

D'Urso, M.G., 2015b, The gravity anomaly of a 2D polygonal body having density contrast given by polynomial functions: Surveys in Geophysics, 36, 391–425.

D'Urso, M.G., 2014a, Gravity effects of polyhedral bodies with linearly varying density: Celestial Mechanics and Dynamical Astronomy, 120(4), 349-372.

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.

Ekinci, Y. L., 2008, 2D focusing inversion of gravity data with the use of parameter variation as a stopping criterion: Journal of the Balkan Geophysical Society, 11 (1), 1-9.

Engen, O., Frazer, L. N., Wessel, P., and Faleide, J. I., 2006, Prediction of sediment thickness in the Norwegian-Greenland Sea from gravity inversion: Journal of Geophysical Research 111, doi:10.1029/2005JB003924.

Eshaghzadeh, A., Sahebari, S.S., and Dehghanpour, A., 2020, 2D inverse modeling of the gravity field due to a chromite deposit using the Marquardt's algorithm and forced neural network: Bulletin of the Mineral Research and Exploration, 161, 33-47.

Eshaghzadeh, A., and Hajian, A., 2018, 2D inverse modeling of residual gravity anomalies from Simple geometric shapes using Modular Feed-forward Neural Network: Annals of Geophysics, 61 (1), SE115. DOI: 10.4401/ag-7540

Essa, K.S., Munschy, M., 2019, Gravity data interpretation using the particle swarm optimisation method with application to mineral exploration: Journal of Earth System Science, 128:123, DOI: 10.1007/s12040-019-1143-4.

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.

Fedi, M., and Mastro, S., 2018, Bounded-region wavelet spectrum: A new tool for depth estimation of gravity and magnetic data: SEG Technical Program Expanded Abstracts, 1425-1429.

Feng, J., Zhang, S., and Meng, X., 2015, Constraint 3D density interface inversion from gravity anomalies: Arabian Journal of Geosciences, 9(1). doi:10.1007/s12517-015-2213-9

Filon, L.N.G. 1928, On a quadrature formula for trigonometric integrals: Proceedings of Royal Society of Edinburgh, 49, 38-47.

Fukushima, T., 2018, Recursive computation of gravitational field of a right rectangular parallelepiped with density varying vertically by following an arbitrary degree polynomial: Geophysical Journal International, 215 (2), 864–879.

Gallardo, L.A., Perez-Flores, M.A., and Gomez-Trevino, E., 2005, Refinement of three-dimensional multilayer models of basins and crustal environments by inversion of gravity and magnetic data: Tectonophysics, 397(1–2), 37–54.

Gallardo-Delgado, L.A., Perez-Flores, M.A., and Gomez-Trevino, E., 2003, A versatile algorithm for joint 3d inversion of gravity and magnetic data: Geophysics, 68(3), 949-959.

García-Abdeslem, J., 2005, The gravitational attraction of a right rectangular prism with density varying with depth following a cubic polynomial: Geophysics, 70(6), J39-J42.

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.

Gerard, A., and Debeglia, N., 1975, Automatic three-dimensional modeling for the interpretation of gravity or magnetic anomalies: Geophysics, 40 (6), 1014–1034.

Gimenez, M.E., Martinez, M.P., Jordan, T, Ruíz, F., and Klinger, F.L., 2009, Gravity characterization of the La Rioja Valley Basin, Argentina: Geophysics, 74(3), B83-B94.

Geng, M., and Yang, Q., 2013, 2-D density inversion with the RBF neural network method: Shiyou Diqiu Wuli Kantan/Oil Geophysical Prospecting, 48(4), 651-657.

Gokula, A.P., and Sastry, R.G., 2015, Gravitational attraction of a vertical pyramid model of flat top-and-bottom with depth-wise parabolic density variation: Journal of Earth System Science, 124(8), 1735-1744.

Goldberg, D.E., 1989, Genetic algorithm: Addison-Wesley Publishing Company.

Gómez-Ortiz, D., Tejero-lópez, R., Babín-Vich, R., and Rivas-Ponce, A., 2005, Crustal density structure in the Spanish Central System derived from gravity data analysis (Central Spain): Tectonophysics, 403, 131-149.

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. and West, G.F., 1965, Interpretation theory in applied geophysics, McGraw-Hill company, New York.

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.

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.

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

Han, D. H., and Batzle, M., 2002, Fizz water and low gas-saturated reservoirs: The Leading Edge, 21 (4), 395-398.

Hansen, R.O., 1999, An analytical expression for the gravity field of a polyhedral body with linearly varying density: Geophysics, 64, 75–77.

Hansen, R.O., and Suciu, L., 2002, Multiple-source Euler deconvolution: Geophysics, 67(2), 525-535.

Hartmann, R.,R., Teskey, D and Friedberg, I., 1971, A system for rapid digital aeromagnetic interpretation: Geophysics, 36, 891-918.

Hood, P., 1965, Gradient measurements in aeromagnetic surveying, Geophysics, 30, 891–802.

Holstein, H., 2003, Gravimagnetic anomaly formulas for polyhedra of spatially linear media: Geophysics, 68 (1), 157–167.

Holstein, H., 2002, Gravimagnetic similarity in anomaly formulas for uniform polyhedral: Geophysics, 67 1126–1133.

Hu, C., Fan, M., Wu, Y., and Pei, Y., 2011, Euler deconvolution method of gravity anomaly based on wavelet analysis: International Conference on Multimedia Technology, Hangzhou, 305-308, DOI: 10.1109/ICMT.2011.6002024.

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.

Hutchison, R.D., 1958, Magnetic analysis by logarithmic curves: Geophysics, 23 (4), 749-769.

Isik, M., and Senel, H., 2009, 3D gravity modeling of Buyuk Menderes basin in western Anatolia using parabolic density function: Journal of Asian Earth Sciences, 34(3), 317–325.

I.T.G.E., 1990, Documentación sobre la geología del subsuelo de España. Tomo V (Duero - Almazán), Internal report number 29040, Madrid.

Jiang, L., Zhang, J., and Feng, Z., 2017, A versatile solution for the gravity anomaly of 3D prism-meshed bodies with depth-dependent density contrast: Geophysics, 82 (4), G77–G86.

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.

Keating, P. B., 1998, Weighted Euler deconvolution of gravity data: Geophysics, 63(5), 1595-1603.

Keating, P., 1992, Density mapping from gravity data using the Walsh transform: Geophysics, 57(4), 637-642.

Kilty, K.T., 1983, Werner deconvolution of profile potential field data: Geophysics, 48 (2), 234–237.

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.

LaFehr, T.R., Nabighian, M.N., 2012, Fundamentals of gravity exploration, Society of Exploration Geophysicists, USA.

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.

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, X., Li, Y., Meng, X., and Luo, Y., 2011, 3-D Inversion of gravity density interface in Hartley domain, SEG Global Meeting Abstracts, 78-78. https://doi.org/10.1190/1.3659121.

Li, X., 2001, Vertical resolution: Gravity versus vertical gravity gradient: The Leading Edge, 20, 901–904.

Li, Y., Oldenburg, D.W., 1998, 3-D gravity inversion of gravity data: Geophysics, 63, 109-119.

Lin, M., and Denker, H., 2019, On the computation of gravitational effects for tesseroids with constant and linearly varying density: Journal of Geodesy, 93, 723-747.

Liu J, Zhang J., Jiang L. Lin, Q. and Wan, L. 2019 Polynomial-based density inversion of gravity anomalies for concealed iron-deposit exploration in North China: Geophysics 84 (5), B325-B334.

Liu, S., Hu, X., Xi, Y., and Liu, T, 2015, 2D inverse modeling for potential fields on rugged observation surface using constrained Delaunay triangulation: Computers & Geosciences, 76, 18-30.

Luo, X., Wang, Y., Zhang, J., Deng, J., Wu, Z., and Chen, S. N., 2018, Applicability analysis and application of Euler deconvolution method in potential field: Computing Techniques for Geophysical and Geochemical Exploration, 40 (6), 789-796.

Mallesh, K., Chakravarthi, V., Ramamma, B., 2019, 3D Gravity Analysis in Spatial Domain: Model Simulation by Multiple Polygonal Cross-Sections coupled with Exponential Density Contrast: Pure and Applied Geophysics, 176(6), 2497-2511,

Manger, G.E., 1963, Porosity and bilk density of sedimentary rocks, U.S. Geological Survey Bulletin, 1144-E, E1-E56.

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., 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.

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.

McCulloh. J. H., 1960, Gravity variation and the geology of the Los Angeles basin of California: U.S.G.S. Prof. Paper 400B, B320-B325.

Medeiros, W.E., and Silva, J.B.C., 1996, Geophysical inversion using approximate equality constraints: Geophysics, 61(6), 1678-1688.

Meinardus, H.A., 1970, On: "Gravity Interpretation using the Fourier Integral" by J. W. Berg, Jr., and M. E. Odegard (Geophysics, 30 (3), 424–438, 1965)": Geophysics, 35(2), 358.

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.

Mohan, N.L., 1978, Interpretation techniques in geophysical exploration using Fourier Transforms: Ph. D thesis, Osmania University, Hyderabad, India (unpublished).

Mohan, N.L., Umashanker, B., and Seshagiri Rao, S.V., 1989, Interpretation of potential field anomalies using the Mellin-Fourier transform: SEG Technical Program Expanded Abstracts, 312-317. https://doi.org/10.1190/1.1889736

Mohan, N.L., Anandababu, and Seshagiri Rao, S.V., 1986, Gravity interpretation using the Mellin transform, Geophysics, 51(1), 114-122.

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.

Mooney, W., Kaban, M., 2010, The North Americal upper mantle: Density, composition, and evolution: Journal of Geophysical research, 115, B12424, DOI:10.1029/2010JB000866.

Moreau, F., Gibert, D., Holschneider, M., and Saracco, G., 1997, Wavelet analysis of potential fields: Inverse Problems, 13, 165-178.

Moreau, F., Gibert, D., Holschneider, M., and Saracco, G., 1999, Identification of sources of potential fields with the continuous wavelet transform - Basic theory: Journal of Geophysical Research, 104, 5003–5013.

Mosegaard, K., Tarantola, A., 1995, Monte Carlo sampling of solutions to inverse problems: Journal of Geophysical Research, 100, 12431-12447.

Mosegaard, K., Tarantola, A., 1991, Monte Carlo analysis of geophysical inverse problems: SEG Technical Program Expanded Abstracts, 940-940.

Mousavi, N., and Ardestani, V.E., 2016, Application of Hyperbolic S-transform in Environmental Gravity Investigation: Geophysics, 21(2), 47-56.

Mundim, K.C., Lemaire, T.J., Bassrei, A., 1998, Optimization of non-linear gravity models through generalized simulated annealing, Physica A, 252, 405–416.

Murthy, I.V.R., 1998, Gravity and Magnetic Interpretation in Exploration Geophysics: Memoir 40, Geological Society of India, 363 p.

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, P. R., Ramakrishna, P., 1989, Gravity anomalies of three dimensional bodies with a variable density contrast: Pure and Applied Geophysics, 30, 711–719.

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, P. R., 1985, A new method of calculating gravity anomalies of three dimensional bodies: Gerl. Beitr. Geophysik, 94, 123-128.

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.

Murthy, I.V.R., 1969, Interpretation of gravity anomalies of dykes by pseudo-magnetic transformation: Pure and Applied Geophysics, 73(1), 43–46.

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.

Nagy, D., 1966, The gravitational attraction of a right rectangular prism: Geophysics, 30, 362–371.

Nasuti, A., and Ardestani, E.V., 2007, 3-D Forward gravity modeling of basement interfaces with quadratic density contrast. EGM 2007 International Workshop. http://www.eageseg.org/data/egm2007/Sessione%20B /Poster%20 papers/ B_PP _07.pdf.

Nelson, T. H., and Fairchild, L., 1989, Emplacement and evolution of salt sills in the northern Gulf of Mexico: Houston Geological Society Bulletin, 32 (1), 6–7.

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.

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.

Ouadfeul, S.A., and Aliouane, L., 2013a, Structural edges delimitation from gravity data using the wavelet transform: Symposium on the Application of Geophysics to Engineering and Environmental Problems, doi:10.4133/sageep2013-142.1

Ouadfeul, S.A., and Aliouane, L., 2013b, Structural edge delimitation from gravity anomaly data using the directional continuous wavelet transform with an example from the Basin and Range Province of the United States: The Leading Edge, 32, 1462-1467.

Pal, P.C., 1983, Rapid gravity interpretation by nomograms: Geoexploration, 21(3), 203-220.

Pal, P.C., 1981a, Gravity profile interpretation using Fourier transform: Geoexploration, 19, 167-177.

Pal, P.C., 1981b, Spectral characteristics of the gravity and magnetic profile data over a geological contact: Geoexploration, 19(3), 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.

Panzner, M., Ebbing, J., and Jordan, M., 2011, 3D gravity inversion constrained by stereotomography: SEG Technical Program Expanded Abstracts.

Paton, S., 1992, Active normal faulting, drainage patterns and sedimentation in southwestern Turkey: Journal of the Geological Society, 149, 1031-1044.

Paul, M.K., 1974, The gravity effect of a homogeneous polyhedron for three-dimensional interpretation: Pure and Applied Geophysics, 112, 553–561.

Pedersen, L. B., 1985, The gravity and magnetic fields from ellipsoidal bodies in the wave number domain: Geophysical Prospecting, 33, 263–281.

Peirce, J.W., and Lipkov, L.,1988, Structural interpretation of the Rukwa Rift, Tanzania: Geophysics, 53, 824-836.

Pham, L. T., Oksum, E., and Do, T.D., 2018, GCH_gravinv: A MATLAB-based program for inverting gravity anomalies over sedimentary basins: Computers & Geosciences, 120, 40-47.

Phillips, J.D., 2014, Using vertical Fourier transforms to invert potential-field data to magnetization or density models in the presence of topography: SEG Technical Program Expanded Abstracts, 1339-1343. https://doi.org/10.1190/segam2014-0226.1

Pilkington, M., Crossley, D.J., 1986, Determination of crustal interface topography from potential fields: Geophysics 51, 1277–1284.

Pohanka, V., 1988, Optimum expression for computation of the gravity field of a homogeneous polyhedral body: Geophysical Prospecting, 36, 733–751.

Prutkin, I., and R. Tenzer, 2009, The optimum expression for the gravitational potential of polyhedral bodies having a linearly varying density distribution: Journal of Geodesy, 83, 1163–1170, doi: 10.1007/s00190-009-0334-1.

Ramamma, B., Mallesh, K., Chakravarthi, V., 2020, 3D Spatial Domain Gravity Inversion with Growing Multiple Polygonal Cross-sections and Exponential Mass Density Contrast: Journal of Earth System Science (under review).

Rao, B.S.R., Murthy, I.V.R., 1978, Gravity and Magnetic methods of prospecting: Arnold-Heinemann publishers (India) Pvt. Ltd., 390 pp.

Rao, K.G.C., Avasthi, D.N., 2006, Analysis of the Fourier spectrum of the gravity effect due to two-dimensional triangular prism, Geophysical Prospecting, 21(3), 526-542.

Rao, P. R., Swamy, K.V., and Murthy, I.V.R., 1999, Inversion of gravity anomalies of three-dimensional density interfaces: Computers & Geosciences, 25, 887-896.

Rao, D.B., and Rao, C.P.V.N.J.M., 1999, Two-dimensional interpretation of gravity anomalies over sedimentary basins with an exponential decrease of density contrast with depth: Proceedings of Indian Academy of Sciences (Earth and Planetary Sciences), 108 (2), 99-106.

Rao, C.V., Chakravarthi, V., and Raju, M. L., 1994, Forward modelling: gravity anomalies of two-dimensional bodies of arbitrary shape with hyperbolic and parabolic density functions: Computers & Geosciences, 20, 873–880.

Rao, D.B., Prakash, M.J., and Babu, N.R., 1993, Gravity interpretation using Fourier transforms and simple geometrical models with exponential density contrast: Geophysics, 58, 1074-1083.

Rao, D.B., Prakash, M.J., 1990a, 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, R.N., 1990b, 3D and $2^{1/2}$D modeling of gravity anomalies with variable density contrast: Geophysical Prospecting, 38, 411–422.

Rao, D.B., 1990c, 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, 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, C. V., and Murthy, I. V. R., 1981, Characteristic curves for interpreting gravity anomalies of vertical cylinders and horizontal circular discs: Proceedings of Indian Academy of Sciences (Earth and Planetary Sciences), 90(3), 197-209.

Rao, V. N., Rao, C. V., and Murthy, I. V. R., 1979, Characteristic curves for interpreting gravity anomalies of two-dimensional vertical prisms of finite depth extent: Geophysical Research Bulletin, 17(1), 37-44.

Rao, B. S. R, and Murthy, I. V. R., 1978, Gravity and magnetic methods of prospecting, Arnold-Heinemann Publishers (India), 390p.

Rao, B. S. R, Murthy, I. V. R., and Rao, C. V., 1974, Gravity interpretation by characteristic curves: Monograph series No. 118, Andhra University, Visakhapatnam.

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.

Regan, R.D., Hinze, W.J., 1977, Fourier transforms of finite length theoretical gravity anomalies: Geophysics, 42(7), 1450-1457.

Regan, R.D., Hinze, W.J., 1978, Theoretical transforms of the gravity anomalies of two idealized bodies: Geophysics, 43(3), 631-633.

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., and Thurston, J.B., 2014, The structural index in gravity and magnetic interpretation: Errors, uses, and abuses: Geophysics, 79(4), J61-J66.

Reid, A.B., Ebbing, J., Webb, S.J., 2014, Avoidable Euler Errors – the use and abuse of Euler deconvolution applied to potential fields: Geophysical Prospecting, 62(5), 1-7.

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.

Ren, Z., Zhong, Y., Chen, C., Tang, J., and Pan, K., 2017, Gravity anomalies of arbitrary 3D polyhedral bodies with horizontal and vertical mass contrasts up to cubic order: Geophysics 83 (1), G1–G13.

Rezaie, M., Moradzadeh, A., and Kalate, A.N., 2017, 3D gravity data-space inversion with sparseness and bound constraints: Journal of Mining & Environment, 8 (2), 227-235.

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.

Roy, Agarwal, Shaw, 2001, A new concept in Euler deconvolution of isolated gravity anomalies: Geophysical Prospecting, 48(3), 559-575.

Ruotoistenmäki, T., 1983, Depth estimation from potential field data using the Fourier amplitude spectrum: Geoexploration, 21(3), 191–201.

Sailhac, P., Gibert, D., and Boukerbout, H., 2009, The theory of the continuous wavelet transform in the interpretation of potential fields - a review: Geophysical Prospecting, 57(4), 517–525.

Salem, A., Masterton, S., Campbell, S., Fairhead, J. D., Dickinson, J., Murphy, C., 2013, Interpretation of tensor gravity data using an adaptive tilt angle method: Geophysical Prospecting, 61(5), 1065-1076.

Sastry, R.G.S., and Moharir, P.S., 1990, Regularising Tunnelling Algorithm in Non-Linear Gravity Problems - a Numerical Study: In: Rummel R., Hipkin R.G. (eds) Gravity, Gradiometry and Gravimetry. International Association of Geodesy Symposia, 103, 171-179. Springer, New York, NY.

Sari, C., Şalk, M., 2002, Analysis of gravity anomalies with hyperbolic density contrast: an application to the gravity data of Western Anatolia: Journal of Balkan Geophysical Society, 5, 87–96.

Sengupta, S., 1977, Interpretation of the gravity effect due to two-dimensional asymmetrical triangular prism by Fourier transforms: Pure and Applied Geophysics, 115(3), 647-654.

Schön, J.H., 1996, Physical Properties of Rocks: Fundamentals and Principles of Petrophysics (Handbook of Geophysical Exploration Series), Pergamon Press, London.

Sharma, B., and Bose, T.K., 1977, A general expression for the Fourier transform of the gravity anomaly due to a fault: Geophysics, 42(7), 1458-1461.

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.

SHELL 1983 Informe del sondeo Baides-1. Internal report.

Shi, Y., 1992, Genetic algorithm and some of its geophysical applications: Acta Geophysica Sinica, 35, 367–371.

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, A., and Biswas, A., 2015, Application of Global Particle Swarm Optimization for Inversion of Residual Gravity Anomalies Over Geological Bodies with Idealized Geometries: Natural Resources Research, 25(3), 297-314.

Skeels, D.C., 1963, An approximate solution of the problem of maximum depth in gravity interpretation: Geophysics, 28(5), 724-735.

Smith, R.A, 1959, Some depth formulae for local magnetic and gravity anomalies: Geophysical Prospecting, 7(1), 55-63.

Smith, R.A, 1960, Some formulae for interpreting local gravity anomalies: Geophysical Prospecting, 8(3), 607-613.

Snyder, W. V., 1978, Contour plotting: ACM Transactions on Mathematical Software, 4, 290-294.

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, 12(3), 373-390.

Stavrev, P., Reid, A., 2010, Euler deconvolution of gravity anomalies from thick contact/fault structures with extended negative structural index: Geophysics, 75, I51-I58.

Storer, D., 1959, Compaction of the argiuaceous sediments in the Padano Basin: In The Gasiferous Deposits of Western Europe, 2, 519–536. Roma, Italy.

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.

Szabo, Z., Pancsics, Z., 1999, Rock densities in the Pannonian basin – Hungary: Geophys Trans 42:5–27.

Talwani, M., and Ewing, M., 1960, Rapid computation of gravitational attraction of three-dimensional bodies of arbitrary shape: Geophysics, 25 (1), 203–225.

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.

Tarantola, A., 2005, Inverse Problem Theory and Model Parameter Estimation: Society for Industrial and Applied Mathematics.

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.

Thompson, D.T., 1982, EULDPH – A new technique for making computer assisted depth estimates from magnetic data: Geophysics, 47, 31–37.

Thorarinsson, F., Magnusson, S.G., and Bjornsson, A, 1988, Directional spectral analysis and filtering of geophysical maps: Geophysics, 53(12), 1587-1591.

Thurston, J., 2010, Euler deconvolution in the presence of sheets with finite widths: Geophysics, 75(3), L71-L78.

Toushmalani, R., 2013, Gravity inversion of a fault by Particle swarm optimization (PSO): Springer Plus, 2, 315.

Tsoulis, D., 2000, A note on the gravitational field of the right rectangular prism: Bollettino di Geodesia e Scienze Affini, 1, 21–35.

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.

Werner, S., 1953, Interpretation of magnetic anomalies at sheet like bodies: Sver. Geol. Undersok. Serv C.. Arsbok, 43, 6.

Wilcox, L.E., 1989, Gravity anomalies interpretation: In The Encyclopedia of solid earth geophysics, David E. James (Ed.), 601.619.

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.

Wu, L., 2019, Fourier-domain modeling of gravity effects caused by polyhedral bodies: Journal of Geodesy, 93, 635–653.

Wu, L., 2018, Comparison of 3-D Fourier forward algorithms for gravity modelling of prismatic bodies with polynomial density distribution: Geophysical Journal International, 215, 1865-1886.

Wu, L., and Chen, L., 2016, Fourier forward modeling of vector and tensor gravity fields due to prismatic bodies with variable density contrast: Geophysics 81(1) G13-G26.

Xia, J., Sprowl, D. R., 1992, Inversion of gravity data with an exponential density contrast model: SEG Technical Program Expanded Abstracts, 525-528. https:// doi.org/10.1190/1.1822137.

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.

Yerkes, R.F., McCulloh, T.H., Schoellhamer, J.E., and Vedder, J.G., 1965, Geology of the Eastern Los Angeles Basin, California – an Introduction: Geological Survey Professional Paper 420-A, United States Government Printing Office.

Yu, P., Wang, J.L., and Wu, J.S., 2007a, An Inversion of Gravity Anomalies by Using a 2.5 Dimensional Rectangle Gridded Model and the Simulated Annealing Algorithm: Chinese Journal of Geophysics, 50(3), 756–764.

Yu, P., Wang, J.L., and Wu, J.S., and Wang, D.W., 2007b, Constrained Joint Inversion of Gravity and Seismic Data Using the Simulated Annealing Algorithm: Chinese Journal of Geophysics, 50(2), 465–475.

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., eds., Special Publication Geological Society of London, 28, 477–493.

Zhang J Z and Jiang L 2017 Analytical expressions for the gravitational vector field of a 3-D rectangular prism with density varying as an arbitrary-order polynomial function: Geophysical Journal International, 210, 1176–1190.

Zhang, J., Zhong, B., Zhou, X., Dai, Y., 2001, Gravity anomalies of 2-D bodies with variable density contrast: Geophysics, 66, 809-813.

Zhang, J., Wang, C-Y., Shi, Y., Cai, Y., Chi, W-C., Dreger, D., Cheng, W-B., and Yuan, Y-H., 2004, Three-dimensional crustal structure in central Taiwan from gravity inversion with a parallel genetic algorithm: Geophysics, 69(4), 917-924.

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 (2010) Analytic solution of the gravity anomaly of irregular 2D masses with density contrast varying as a 2D polynomial function. Geophysics 75:I11–I19.

Zhou, X., 2008, 2D vector gravity potential and line integrals for gravity anomaly caused by a 2D mass of depth-dependent density contrast: Geophysics, 73, I43–I50.

Zhdanov, M. S., and Cai, H., 2013, Inversion of gravity and gravity gradiometry data for density contrast surfaces using Cauchy-type integrals: SEG Expanded Abstracts, https://doi.org/10.1190/segam2013-0429.1.

IOP Publishing Journal of Geophysics and Engineering
J. Geophys. Eng. 14 (2017) 303–315 (13pp) https://doi.org/10.1088/1742-2140/aa5832

Basement depth estimation from gravity anomalies: two 2.5D approaches coupled with the exponential density contrast model

V Chakravarthi[1], **K Mallesh and B Ramamma**

Centre for Earth and Space Sciences, University of Hyderabad, Gachibowli, Hyderabad—500046, Telangana, India

E-mail: vcvarthi@rediffmail.com, malleshkamal08@gmail.com and ramageophd@gmail.com

Received 20 July 2016, revised 23 November 2016
Accepted for publication 10 January 2017
Published 13 February 2017

Abstract

We develop two automatic techniques in the spatial domain using the exponential density contrast model (EDCM) to trace the bottom surface of a 2.5D sedimentary basin from the observed gravity anomalies. The interface between the sediments and basement is described with a finite strike polygonal source, whose depth ordinates become the unknown parameters to be estimated. The proposed automatic modeling technique makes use of the forward difference approximation and the inversion solves a system of normal equations using the ridge regression to estimate the unknown parameters. Furthermore, the proposed inversion technique simultaneously estimates the regional gravity background that is associated with the residual gravity anomaly. In either case, forward modeling is realized in the spatial domain through a method that combines both analytical and numerical approaches. The utility of each algorithm was successfully tested on a theoretically produced noisy residual gravity dataset. The validity of the inversion technique is also exemplified with the noisy gravity anomalies attributable to a synthetic structure in the presence of regional gravity background. We demonstrate that the magnitude of gravity anomaly is offset dependent and that it would influence the modeling result. Additionally, some applications with real gravity datasets from the Gediz and Büyük Menderes grabens in western Turkey using the derived EDCMs have produced geologically reasonable results which are in close agreement with those reported previously.

Keywords: gravity anomalies, basement interfaces, 2.5D, modeling, inversion, exponential density contrast model

(Some figures may appear in colour only in the online journal)

Introduction

One of the major applications of the gravity method is to decipher the concealed basement geometries under the sedimentary load. In thick sedimentary basins electric and electromagnetic methods cannot provide information at sufficient depth, whereas, gravity data yields valuable information on the basement (Martelet *et al* 2013). Usually, negative gravity anomalies are observed over the sedimentary basins because the density of sedimentary rocks is less than the surrounding basement. These anomalies can be analyzed using appropriate modeling and inversion schemes to decipher the basement configurations.

Undoubtedly, a thorough knowledge of the density of rocks both at the surface and subsurface is always essential for applying a few gravity corrections to the gravity measurements and also to obtain reliable geologic interpretations of the gravity anomalies. However, in case of sedimentary rocks the density is rarely uniform (Kinsman 1975, Rusakov 1990, Ramillien and Wright 2002, Braitenberg *et al* 2006, Klinger *et al* 2011, Azab and El-Khadragy 2013).

Athy (1930) was the first researcher to express the relations between the depth of burial and the density, porosity,

[1] Author to whom any correspondence should be addressed.

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 efficiently by exponential density-depth relationship. Cowie and Karner (1990) have demonstrated from the measured density-depth data of different stratigraphic units in 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. 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. Based on the analysis of the density samples taken from 716 drill sites of the Deep Sea Drilling Project, Tenzer and Gladkikh (2014) showed that the density increases nonlinearly with the increasing sediment depth due to compaction.

Cai and Zhdanov (2015) argue in line with Cordell (1973) 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. From the actual measurement of sedimentary rock density from deep bore holes, Cordell (1973) had established that the density contrast decreases drastically at shallow depths and less progressively at deeper depths following an exponential law. Although several mathematical functions are in use to describe the density contrast variation of sedimentary rocks with depth viz., linear (Pedersen 1985, Reamer and Ferguson 1989, Hansen 1999, Hamayun *et al* 2009, D'Urso 2014), quadratic (Bhaskara Rao 1986, 1990, Gallardo-Delgado *et al* 2003), cubic polynomial (Garcia-Abdeslem 2005); each one has its own limitations in its application as demonstrated by Chakravarthi and Sundararajan (2006) and Chakravarthi (2009). Hence, the choice of exponential density contrast model (EDCM) in the analysis of gravity anomalies of sedimentary basins is more appropriate to obtain reliable interpretations.

However, the major intricacy associated with the EDCM is that no closed form analytical expressions for the gravity anomalies could be derivable in the spatial domain for forward modeling (Radhakrishna Murthy 1998, Chakravarthi and Sundararajan 2004). Owing to this difficulty, methods have been proposed in the spectral domain to realize forward modeling. For e.g., Cordell (1973) had developed a recursive method combining 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, Granser (1987) proposed a forward modeling algorithm based on Taylor series expansion to calculate the gravity anomalies of sedimentary basins, Bhaskara Rao *et al* (1993) developed a few graphical methods to analyze the Fourier transforms of the gravity anomalies of simple 2D geometric models. Bhaskara Rao and Mohan Rao (1999) proposed a method based on the Bott's (1960) approach to analyze the gravity anomalies of 2D sedimentary basins, where forward modeling was performed in the spectral domain followed by their transformation to the space domain by Filon's (1928) method. Chai and Hinze (1988) also calculated anomalies of prismatic bodies in the wave number domain and converted the anomalies back to the spatial domain by applying a shift-sampling technique. Chappel and Kusznir (2008) derived wave number domain expressions to calculate the gravity anomalies of sedimentary basins with irregular bounding surfaces. All the above-mentioned techniques, except the ones reported by Granser (1987) and Chai and Hinze (1988), are 2D. The methods proposed by Granser (1987) and Chai and Hinze (1988) are applicable to analyze either profile data or two dimensional data. Nevertheless, truncation errors would crop up in all the enlisted methods when the gravity anomalies transform back to the spatial domain from frequency domain (Chakravarthi and Sundararajan 2007, Chakravarthi *et al* 2016).

In some cases, sedimentary basins that are formed particularly due to strike slip motions (for e.g., pull-apart basins) more often posses finite strike lengths; hence, the anomalous mass needs to be approximated by limited/finite strike models to analyze the gravity anomalies produced by them. Nevertheless, 3D models are more expensive in terms of both data requirement and computational time than 2D; therefore, use of 2.5 dimensionality (2.5D) in the quantitative interpretation of gravity anomalies is justified.

In this context, Rama Rao and Radhakrishna Murthy (1989) have developed two forward modeling schemes coupled with relevant codes to compute the gravity anomalies of 2D and 2.5D polygonal sources using uniform or linear or exponential density variations. To accommodate linear or exponential density variation they proposed subdivision of each side of the polygon into segments for e.g., 1 for the linear density and 10 for the exponential density. This technique, though efficient, invariably consumes significant amount of time (Visweswara Rao *et al* 1994, Chakravarthi *et al* 2016). Mickus and Peeples (1992) have devised a technique based on the inverse theory of Backus and Gilbert (1967, 1968, 1970) to trace the bottom surface of a 2.5D sedimentary basin from the observed gravity and magnetic fields. However, this technique finds limited practical application particularly in cases where the geological settings warrant the use of variable density. In recent past, a 2.5D inversion technique coupled with EDCM to estimate the basement depths from the observed gravity anomalies was developed by Chakravarthi *et al* (2013); wherein the sedimentary pile above the basement was described with a collage of vertical prisms having finite but variable strike lengths. However, this technique is difficult to implement if the observed gravity anomalies are available at random intervals along a profile. On the other hand, the interpretation algorithms of Chakravarthi *et al* (2016) are free from the above

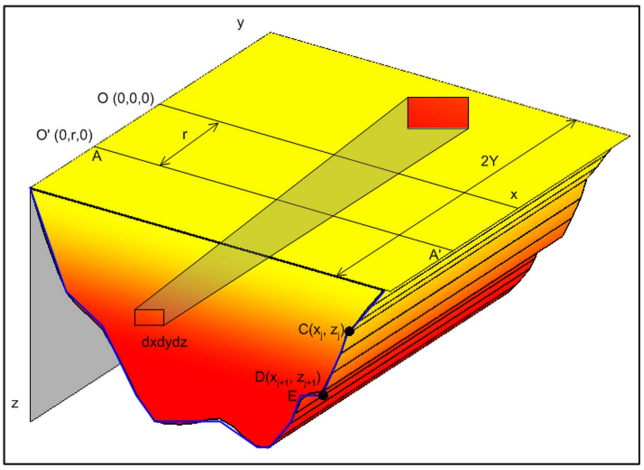


Figure 1. Schematic representation of the depth structure of a finite strike sedimentary basin (solid line in black) and its approximation by a 2.5D finite strike polygonal source (solid line in blue). The finite strike, $2Y$, of the basin prevents one to represent it by a 2D source. Here, r, is the offset distance of a selected profile, AA'.

constraint, but again these are 2D. Above all, the enlisted methods are strictly valid for residual gravity anomalies alone.

In the present paper, two automatic techniques in the spatial domain are developed; one based on the principles of automatic modeling and the other on inversion (Chakravarthi *et al* 2016) to estimate the basement depths from a set of randomly distributed gravity anomaly data at any profile offset considering (i) finite strike length for a sedimentary basin, and (ii) exponential density contrast variation within the sedimentary pile. The proposed automatic modeling technique uses the forward difference approximation and the inversion technique solves the system of normal equations to estimate the unknown parameters. In case of automatic modeling, the anomalies attributable to a synthetic model in the presence of pseudorandom noise are analyzed and in case of inversion the noisy anomalies are analyzed both with and without regional gravity background. In either case the estimated parameters are compared with the assumed ones. To demonstrate the practical applicability of the proposed techniques three real field gravity anomaly profiles from western Turkey are interpreted and judged against the interpretations previously reported (Sari and Şalk 2002).

Forward modeling—theoretical considerations

Figure 1 shows the depth structure of a typical finite strike sedimentary basin in the xz-plane of the Cartesian co-ordinate system. The basin possesses uniform cross-section throughout its strike length ($2Y$) along the y-axis perpendicular to the xz-plane. Let the z-axis be positive vertically downwards along which the depth dimension of the basin is scaled and x-axis runs transverse to the strike of the basin. The gravity anomaly of the basin at any observation, $O'(0, r, 0)$, on the xy-plane can be obtained by integrating the gravity effect of an element throughout the volume of the basin given by

$$\Delta g_{2.5\mathrm{D}}(0,\, r,\, 0) = G \int_v \frac{\Delta\rho(z) z \, \mathrm{d}x\mathrm{d}y\mathrm{d}z}{(x^2 + \overline{y - r}^2 + z^2)^{3/2}}, \qquad (1)$$

where G is universal gravitational constant, (x, y, z) are source coordinates, $\mathrm{d}x\mathrm{d}y\mathrm{d}z$ is the volume of an element within the source, and r is the offset of the profile, AA', from the origin $O(0, 0, 0)$. $\Delta\rho(z)$ represents EDCM at any depth z within the structure given by Cordell (1973)

$$\Delta\rho(z) = \Delta\rho_0 \mathrm{e}^{-\lambda z}. \qquad (2)$$

Here $\Delta\rho_0$ is the density contrast observed at the ground surface and λ is a decay factor expressed in reciprocal length units. Upon substitution equation (2) and performing partial integration with respect to y, equation (1) becomes

$$\Delta g_{2.5\mathrm{D}}(0,\, r,\, 0) = G\Delta\rho_0 \int_s \frac{z\mathrm{e}^{-\lambda z}}{(x^2 + z^2)} \times \left[\frac{Y - r}{\sqrt{(x^2 + \overline{Y - r}^2 + z^2)}} + \frac{Y + r}{\sqrt{(x^2 + \overline{Y + r}^2 + z^2)}} \right] \mathrm{d}x\mathrm{d}z, \qquad (3)$$

where s stands for surface integration. Applying Stokes' theorem, equation (3) takes the form

$$\Delta g_{2.5\mathrm{D}}(0,\ r,\ 0) = G\Delta\rho_0 \oint_z \mathrm{e}^{-\lambda z} \times \left[\arctan \frac{x\,\overline{Y - r}}{z\sqrt{(x^2 + \overline{Y - r}^2 + z^2)}} + \arctan \frac{x\,\overline{Y + r}}{z\sqrt{(x^2 + \overline{Y + r}^2 + z^2)}} \right] \mathrm{d}z. \qquad (4)$$

Approximating the outline of the basin by a polygon CDE… (shown as a solid line in blue in figure 1), the x term in equation (4) can be expressed for the Jth side, such as CD as

$$x = a + z\cot i, \qquad (5)$$

where $a = x_j - z_j \cot i$ and i is the angle made by the side CD with the x-axis. Here, (x_j, z_j) are the coordinates of the vertex C. Upon substitution of equation (5) in equation (4) the gravity effect of the Jth side of the polygon can be obtained as

$$\Delta g_{\mathrm{CD}}(0,\, r,\, 0) = G\Delta\rho_0 \oint_{z_j}^{z_{j+1}} \mathrm{e}^{-\lambda z} \times \left[\arctan \frac{(a + z\cot i)\,\overline{Y - r}}{z\sqrt{(\overline{a + z\cot i}^2 + \overline{Y - r}^2 + z^2)}} + \arctan \frac{(a + z\cot i)\,\overline{Y + r}}{z\sqrt{(\overline{a + z\cot i}^2 + \overline{Y + r}^2 + z^2)}} \right] \mathrm{d}z. \qquad (6)$$

Here z_{j+1} is the depth ordinate of the vertex D, whose coordinate is represented by x_{j+1}. The total gravity effect of the

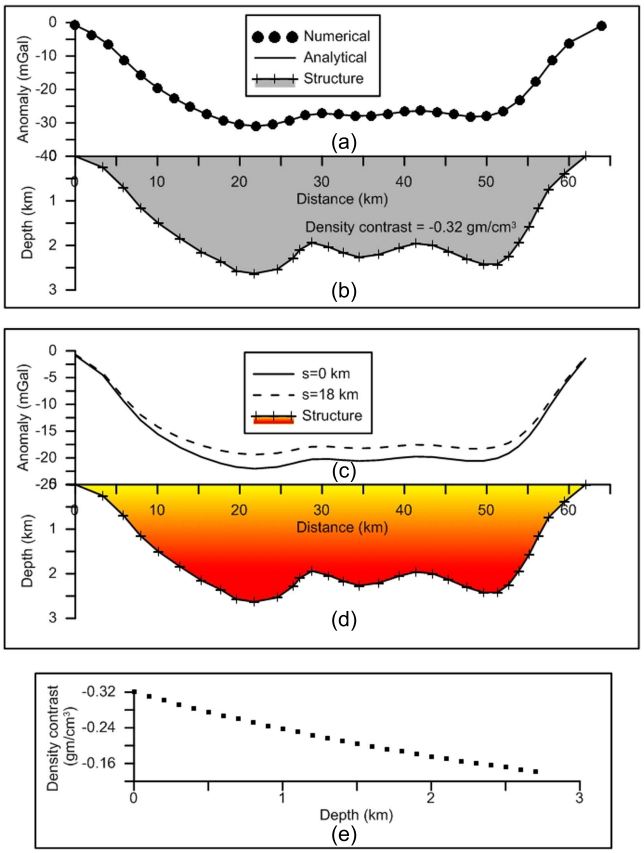


Figure 2. (a) Gravity anomalies calculated at zero offset by analytical (Radhakrishna Murthy 1998) and proposed methods using uniform density contrast, (b) geometry of a finite strike sedimentary basin, (c) gravity anomalies at 0 and 18 km offsets with exponential decrease in density contrast, (d) geometry of a basin in which the color gradation from yellow to red indicates the increase in density of sediments with depth, (e) prescribed exponential density contrast model used in forward modeling.

basin at the observation, $O'(0, r, 0)$, can be obtained as

$$\Delta g_{2.5\mathrm{D}}(0, r, 0) = \sum_{j=1}^{N} \Delta g_j(0, r, 0), \tag{7}$$

where N stands for the number of sides of the polygon. It is to note that closed form analytic solution does not exist for equation (6) in the spatial domain; hence, necessity arises to solve the equation by a numerical approach. The vertices of the anomalous source are covered sequentially in the clockwise direction.

In the presence of regional background, Ψ, the gravity anomaly of the basin can be represented as

$$\Delta g_{\mathrm{total}}(0, r, 0) = \Delta g_{2.5\mathrm{D}}(0, r, 0) + \Psi, \tag{8}$$

where

$$\Psi = \sum_{i=0}^{N1} f_i x^i. \tag{9}$$

Here $N1$ stands for the degree of polynomial and x is the observer location on the profile.

We demonstrate the validity of the proposed numerical method of anomaly calculation on a finite strike synthetic model, the geometry of which is shown in figure 2(b). In this case, we presume the length of the model as 40 km along the strike (half-strike length is 20 km) with 32 irregularly spaced vertices describe the geometry of the undulating density interface (figure 2(b)).

Presuming that the sedimentary basin is homogeneous throughout its volume with a density contrast of $-0.32\ \mathrm{gm\,cm^{-3}}$, the residual gravity anomalies calculated from 32 unequally spaced observations in the interval $x_j \in [0\ \mathrm{km}, 64\ \mathrm{km}]$ using equation (7) (shown as solid dots in figure 2(a)) are compared with the anomalies obtained from an analytical solution (shown as a solid line in figure 2(a)) of Radhakrishna Murthy (1998). It is to note that the x-coordinates of the vertices of the polygon in this case do not coincide with the observer locations on the profile. The fact that the error between the two anomalies barely exceeds $\pm 1 \times 10^{-3}$ mGal confirms the reliability and accuracy of the proposed numerical method.

Further, figure 2(c) shows the gravity anomalies produced by a structure (figure 2(d)) along two selected profiles at two different offsets 0 and 18 km; one each in the interval $x_j \in [0\ \mathrm{km}, 62\ \mathrm{km}]$. It is to note that the depth structure shown in figure 2(d) is exactly the same as the one shown in figure 2(b) but in this case the density contrast within the structure obeys exponential decrease with depth (figure 2(e)). The surface density contrast and the decay constant of EDCM (equation (2)) are presumed as $-0.32\ \mathrm{gm\,cm^{-3}}$ and $0.3\ \mathrm{km^{-1}}$ respectively. In this case, distances to the observer locations on the profile form the x-coordinates of the vertices of the polygon. The magnitudes of the gravity anomalies calculated at two different offsets are different over the lengths of the profiles (figure 2(c)), though the structure remains the same (figure 2(d)). For e.g., the model produces a maximum gravity anomaly (absolute) of 22 mGal at the 22nd km on the profile at zero offset; whereas it is 19 mGal at 18 km offset (figure 2(c)). It is to realize that even a 3 mGal difference in the anomaly would seriously affect the interpretation if appropriate offset is not chosen in the analysis.

Analysis of gravity anomalies

In general, interpretation of gravity anomalies is carried out by specifying a set of approximate depths to the density interface supplemented from known geologic information/drilling and/or other geophysical inputs. The gravity response of the structure is then calculated with a suitable forward modeling algorithm and compared with the observed anomaly. The difference between the two anomalies is minimized by adjusting the depths to the interface within the permissible limits based on some convergence criteria. For optimum depth estimates the model gravity response should mimic the observed response and the corresponding estimated structure is geologically sensible.

If the profile along which the interpretation is intended covers the lateral dimensions of a sedimentary basin completely then the relief of the basin at the first and last observations become zero. If the distances to observer locations on a profile and the x-coordinates of the vertices of the polygon

are the same then one needs to estimate $(N-2)$ depth parameters from N observed anomalies. The present automatic modeling and inversion algorithms initiate the structure of a sedimentary basin presuming that the observed gravity anomaly at each observation on the profile is being produced by an infinitely extending horizontal slab in which the density contrast decreases with depth following equation (2). According to Cordell (1973), the thickness of such a slab can be estimated as

$$z_{B_i} = \frac{-1}{\lambda}\log\left(1 - \frac{\lambda g_{B_i}}{2\pi G\Delta\rho_0}\right), \tag{10}$$

where, g_{B_i} is the gravity anomaly observed at any station, i (x_i). Because the width of a sedimentary basin is always finite across its strike, the depth (thickness) estimates realized from equation (10) are only approximate; thereby, the model gravity anomalies calculated from equations (6) to (7) obviously differ in magnitude from the observed anomalies, $\Delta g_{\text{obs}}(x_i)$. The difference between the observed, $\Delta g_{\text{obs}}(x_i)$, and model gravity response, $\Delta g_{2.5\text{D}}(x_i)$, at all observations can be quantified by a root mean square error (Chakravarthi *et al* 2016) defined by

$$J_{\text{rms}} = \sqrt{\frac{\sum_{i=1}^{N}[\Delta g_{\text{obs}}(x_i) - \Delta g_{2.5\text{D}}(x_i)]^2}{N_{\text{obs}}}}. \tag{11}$$

Here $N = N_{\text{obs}}$. In case of automatic modeling, the basin depths are improved based on the forward difference approximation

$$z_{B_i}{}^{k+1} = z_{B_i}{}^{k} + \Delta z_{B_i}{}^{k}, \tag{12}$$

where

$$\Delta z_{B_i}{}^{k} = \frac{-1}{\lambda}\log\left(1 - \frac{\lambda[\Delta g_{\text{obs}}(x_i) - \Delta g_{2.5\text{D}}(x_i)]}{2\pi G\Delta\rho_0 \mathrm{e}^{-\lambda z_{B_i}{}^{k}}}\right). \tag{13}$$

Here k represents the number of iterations. It is to note that in case of automatic modeling the gravity anomalies attributable solely to a sedimentary basin are used to estimate the basement depths, whereas in case of inversion the residual gravity anomalies may be associated with regional background (Radhakrishna Murthy 1998, Chakravarthi *et al* 2013). The proposed inversion algorithm estimates the regional background in addition to the depth parameters of the interface. The modified depths obtained from equation (13) are used to update the gravity response of the structure and a new rms error, J_{rms1} is calculated. If the magnitude of J_{rms1} is less than J_{rms} then the value of J_{rms1} is assigned to J_{rms} and $z_{B_i}{}^{k+1}$ to $z_{B_i}{}^{k}$ and the process continues till (i) the specified number of iterations completed, or (ii) the resulting rms error becomes less than the predefined allowable error, or (iii) the new rms error exceeds its preceding value.

In case of inversion, the number of unknown parameters to be estimated from $N(=N_{\text{obs}})$ observed anomalies is $(N-2)+N1+1$. Although one may choose any degree of polynomial (equation (9)) to describe the regional background and then to realize forward modeling using equation (8), it is always necessary to restrict the unknown parameters to be estimated from a given set of observed data to a number less than or equal to the number of observations. For this reason, the regional background in this case needs to be presumed either constant throughout the length of the profile or would vary linearly as a function of observer location. In the present inversion technique we have assumed that the regional background varies linearly along the profile for which case the number of unknown parameters to be estimated becomes N, being equal to the number of observations. In case higher order polynomials are required to simulate the regional background, the number of depth parameters to be solved shall be cut down appropriately by means of using known basement depths as constraints in the inversion.

To start with the coefficients f_0 and f_1 of the polynomial (equation (9)) are set to zero and subsequently updated iteratively along with the depth parameters of the basement. The difference between the observed $(\Delta g_{\text{obs}}(x_i))$ and model gravity anomaly $(\Delta g_{\text{total}}(x_i))$ at any observation, x_i, at the end of kth iteration can be expressed as

$$\Delta g_{\text{obs}}(x_i) - \Delta g_{\text{total}}(x_i) = \sum_{j=2}^{N-1}\frac{\partial\Delta g(x_i)}{\partial z_i}\mathrm{d}z_i + \mathrm{d}f_0 x_i + \mathrm{d}f_1. \tag{14}$$

The N normal equations that are constructed and solved to estimate the improvements in N unknown parameters by minimizing the rms error (equation (11)) using the ridge regression algorithm (Marquardt 1970) can be expressed in a matrix form as

$$(A + \delta I)X = B, \tag{15}$$

where A is a $N \times N$ matrix, whose elements $A_{nj'}$ are given by

$$A_{nj'} = \sum_{n=1}^{N}\sum_{m=1}^{N}\frac{\partial\Delta g(x_m)}{\partial a_{j'}}\frac{\partial\Delta g(x_m)}{\partial a_n}, j' = 1,\ 2,\ldots,N, \tag{16}$$

$$X = \mathrm{d}a_n, \tag{17}$$

$$B = \sum_{m=1}^{N}[\Delta g_{\text{obs}}(x_m) - \Delta g_{\text{total}}(x_m)]\frac{\partial\Delta g(x_m)}{\partial a_{j'}},$$
$$j' = 1,\ 2,\ldots,\ N. \tag{18}$$

Here, $a_n = z_n,\ n = 1,\ \ 2, \ldots, N-2,$

$$a_{N-1} = f_0$$

and

$$a_N = f_1.$$

Also, δ is the damping factor and I is a diagonal matrix containing the diagonal elements of the matrix A. The partial derivatives required in equations (16) and (18) are evaluated numerically following the procedure described by Chakravarthi *et al* (2013). The application of the ridge regression algorithm is detailed by Chakravarthi (2003) and Chakravarthi *et al* (2016). The estimated improvements in the unknown parameters, $\mathrm{d}a_n$, solved from equation (15) are added to/subtracted from the existing parameters, a_n, $n = 1,\ 2,\ \ldots,\ N$ and the inversion process continues for the

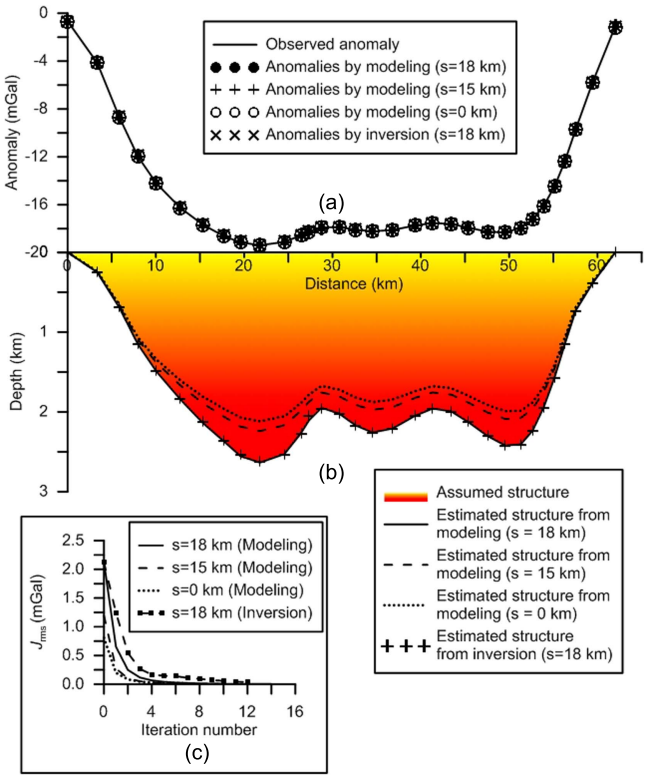


Figure 3. (a) Observed and theoretical anomalies by automatic modeling and inversion at different offsets in the presence of pseudorandom noise, (b) assumed and estimated structures by modeling and inversion for different offsets, (c) changes in rms errors with iteration for different offsets.

specified number of iterations or until one of the following criteria is fulfilled

(i) the resulting misfit becomes less than the predefined allowable error, or
(ii) the resulting damping factor attains an unusually large value (Chakravarthi 2003).

Applications

The applicability of each proposed interpretation methodology is established by analyzing the gravity anomalies produced by a synthetic structure before being applied to analyze the real world gravity anomalies.

Synthetic example

The gravity anomalies shown in figure 2(c), which are produced by a structure at 18 km offset, are treated as the observed anomalies for modeling purpose. These anomalies are further corrupted with pseudorandom noise (Gaussian) having zero mean and a standard deviation of 0.1 mGal before the analysis attempted. The noisy anomalies of the structure are shown in figure 3(a) as a solid line in black. Automatic modeling and inversion techniques, described in the text, are applied on these noisy anomalies to recover the structure by

Table 1. Assumed and estimated coefficients of linear regional background, synthetic example.

	Assumed		Estimated	
Offset (km)	f_0(mGal)	f_1(mGal km^{-1})	f_0(mGal)	f_1(mGal km^{-1})
18	−0.33	−0.023	−0.429	−0.016
15			−0.365	−0.015
0			−0.302	−0.0143

setting different values to the offset parameter independently (18, 15 and 0 km) to study its effect on the interpretation, if any. In a real case, the strike length of a basin needs to be established either from existing geological and/or Bouguer anomaly maps. The offset of the profile along which interpretation is intended can be easily found. The values of $\Delta\rho_0$ and λ of EDCM (equation (2)) remain unchanged during the process of analysis in both cases.

When the offset was set to 18 km, the modeling algorithm has performed 15 iterations before it got terminated. For the same anomaly and chosen offset the inversion took 12 iterations. The rms error (equation (11)) for the starting model in either case was 2.09 mGal. In case of modeling, the error was drastically reduced to 0.13 mGal at the end of the 3rd iteration, beyond which it showed a progressive decay with the iteration number (figure 3(c)). On the other hand, in case of inversion the decay of rms error was found to be rather gradual when compared to modeling. The optimum estimated structure corresponding to an rms error of 0.05 mGal (figure 3(c)) between the observed and modeled gravity anomalies was obtained at the end of the 15th iteration in case of modeling and 12th iteration in case of inversion (figure 3(b)). The theoretical gravity anomalies obtained from both modeling and inversion are shown in figure 3(a) with the corresponding estimated structures in figure 3(b). One can notice from figure 3(b) that the estimated structure from modeling exactly mimics the structure obtained from inversion.

On the other hand, when the algorithms are applied on the same anomaly with 15 km offset; modeling took 10 iterations and inversion 12 iterations for a proper convergence. On the other hand, for 0 km offset modeling performed 7 iterations and inversion 11 iterations. Beyond the concluding iterations the modeling technique was terminated because the resulting rms error in each case exceeds its preceding value, whereas in case of inversion the damping factor attained a large value. The changes in rms errors associated with the improvements in model space at both offsets from modeling are shown in figure 3(c). The theoretical gravity anomalies and the estimated depth structures from modeling for $s = 0$ km and $s = 18$ km offsets are shown in figures 3(a) and (b) for comparison. The model gravity responses and inferred depth structures realized from the inversion technique for both the offsets (0 and 15 km) are exactly the same as the ones obtained from modeling; hence are not shown in figures 3(a) and (b) for brevity.

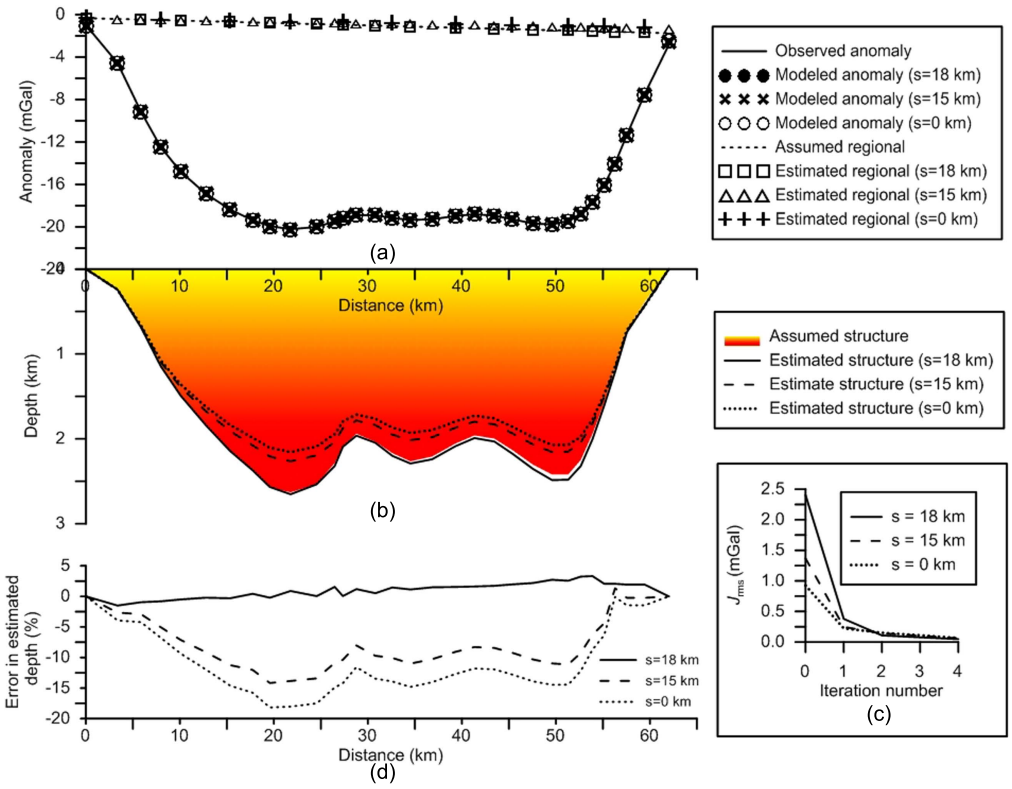


Figure 4. (a) Observed and theoretical anomalies by inversion at different offsets in the presence of both regional background and pseudorandom noise, (b) assumed and estimated structures by inversion for different offsets, (c) changes in rms errors with iteration for different offsets, (d) error (%) between the estimated and assumed depths for different assumed offsets.

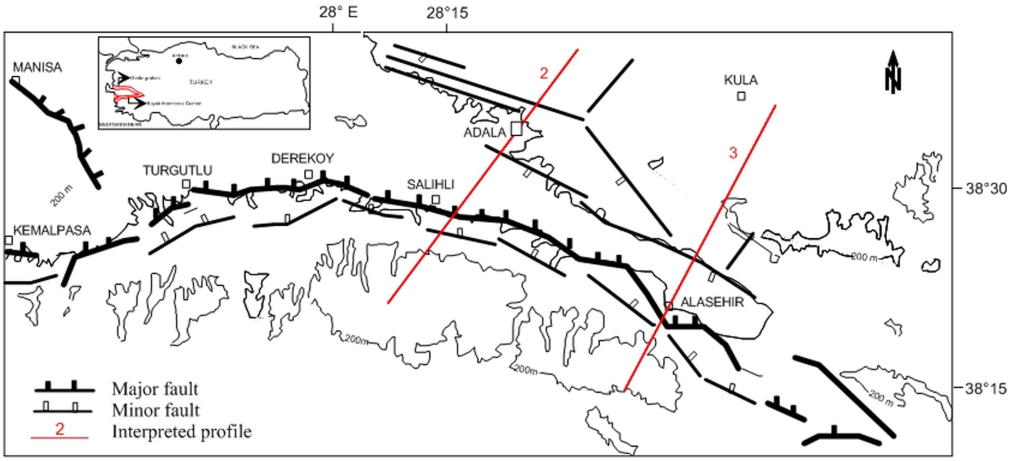


Figure 5. Faulting along Gediz graben, western Anatolia. Main faults are shown in thick lines with solid ticks on downthrown side; minor faults are in thinner lines with open ticks on downthrown side (reproduced from Sari and Şalk 2002, Chakravarthi and Sundararajan 2007). Gravity anomalies along the Profiles 2 and 3 are interpreted.

It can be seen from figure 3(a) that the theoretical gravity responses realized from both modeling and inversion at different offsets equally explain the observed anomaly but the inferred depth structures are dissimilar (figure 3(b)). The structures deciphered by setting the offset to 18 km closely mimic the assumed one, where as the other two structures obtained with zero and 15 km offsets show undue deviations. It is observed that as the magnitude of the offset decreases the resulted structure obtained from either modeling or inversion becomes more and more underestimated (figure 3(b)). A few insignificant deviations in the estimated structures are inevitable (around 30th km) even considering an appropriate offset of 18 km in the interpretation, however, such deviations can be ignored because the anomalies used in the analysis are noisy.

To study the combined effect of both regional background and pseudorandom noise on the interpretation, regional background described with a set of predefined

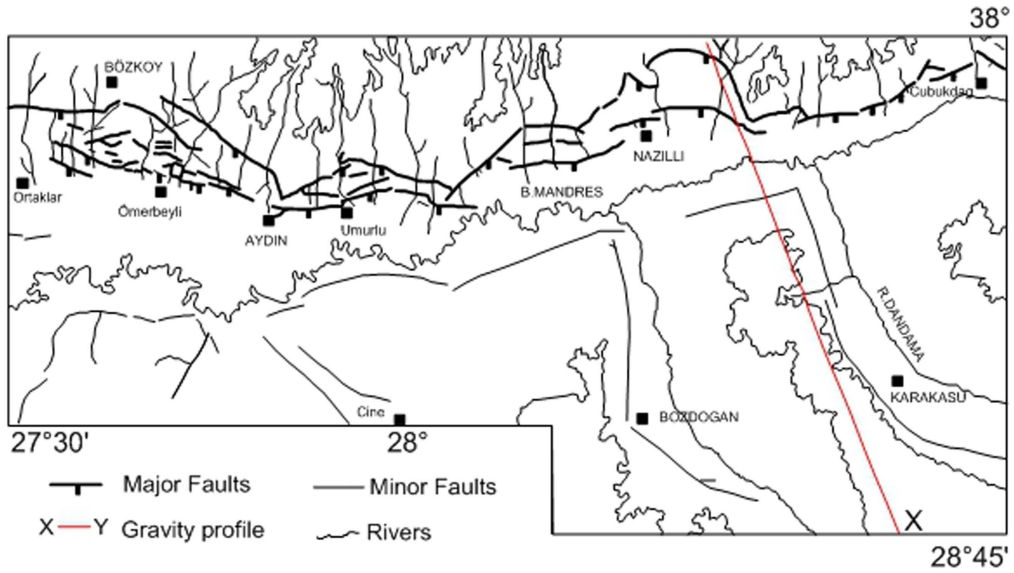


Figure 6. Faulting along Büyük Menderes graben, western Anatolia. Main faults are shown in thick lines with solid ticks on downthrown side; minor faults are in dashed lines (reproduced from Sari and Şalk 2002). Gravity anomalies along the Profile XY are interpreted.

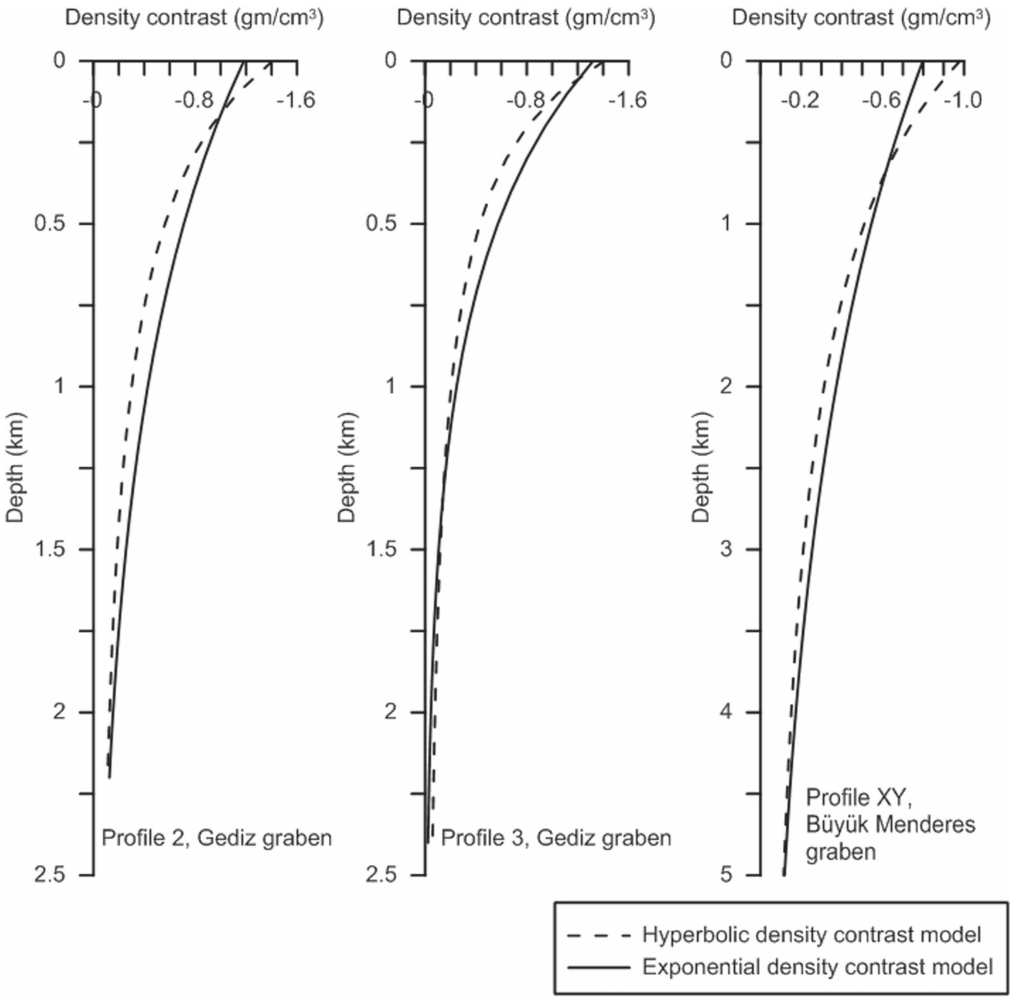


Figure 7. Derived hyperbolic (reproduced from Sari and Şalk 2002) and exponential density contrast models, Gediz and Büyük Menderes grabens, western Turkey.

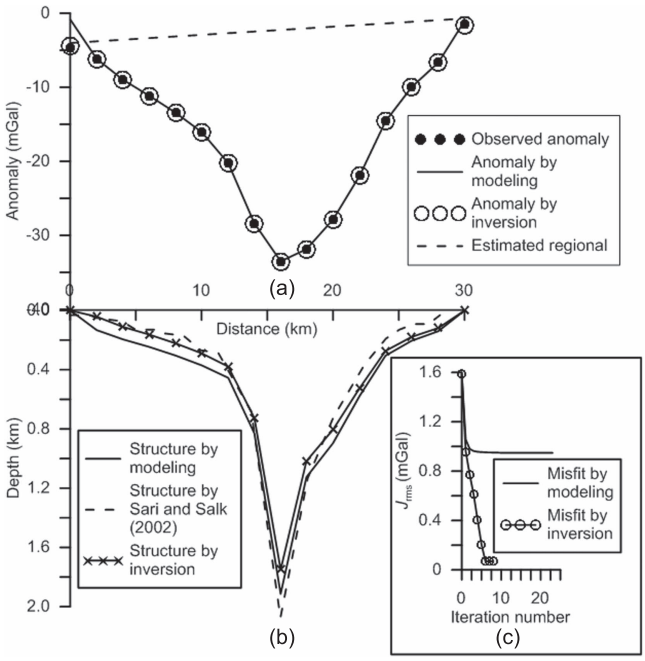


Figure 8. (a) Observed and theoretical gravity anomalies by automatic modeling and inversion along Profile 2, Gediz graben, western Turkey. Estimated regional is also shown, (b) inferred depth structures of the graben by automatic modeling and inversion. Estimated structure by Sari and Şalk (2002) based on 2D modeling is also shown for comparison, (c) changes in rms errors with iteration.

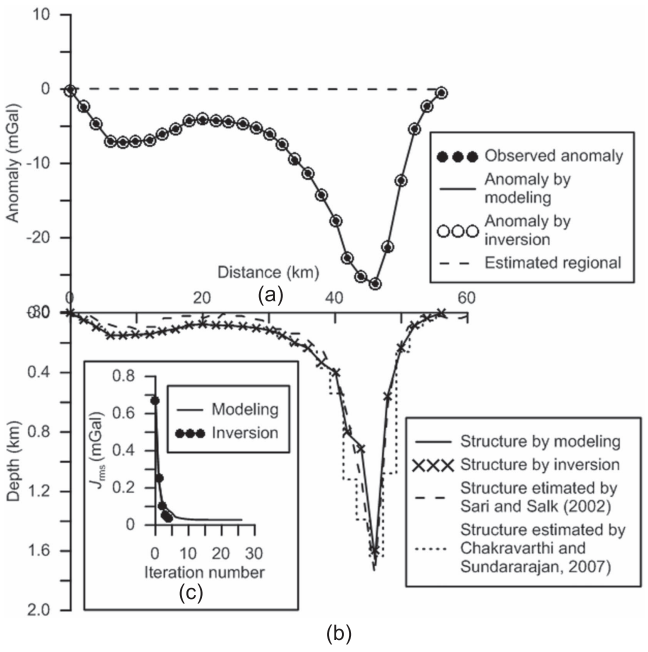


Figure 9. (a) Observed and theoretical gravity anomalies by automatic modeling and inversion along Profile 3, Gediz graben, western Turkey. Estimated regional is also shown, (b) inferred depth structures of the graben by automatic modeling and inversion. Estimated structures by Sari and Şalk (2002), and Chakravarthi and Sundararajan (2007) are also shown for comparison, (c) changes in rms errors with iteration.

coefficients (table 1), by setting $N1 = 1$ in equation (9), is added to the noisy anomalies shown in figure 3(a) and subsequently inverted to recover the basement structure. The assumed regional field and the regional associated noisy anomalies are shown in figure 4(a).

The analysis was performed on the anomalies independently, as in the previous case, by setting the offsets to 18 km, 15 km and 0 km respectively. In each case, the inversion algorithm has performed four iterations and then terminated because the resulting rms error attained a value less than the predefined allowable error (figure 4(c)). The theoretical gravity anomalies at the end of the concluding iteration closely mimic the observed anomalies in all cases (figure 4(a)); whereas the corresponding estimated structures show deviations from each other (figure 4(b)).

The deciphered structure with 18 km offset was slightly overestimated in the range $x_j \in [30\text{ km}, 55\text{ km}]$ with a maximum error of 3.3% found at the 54th km (figure 4(d)). On the other hand, when the offset was set to 15 and 0 km the algorithm in each case has yielded a structure that was underestimated considerably over the length of each profile. With 15 km offset a maximum error of 14.2% in the estimated depth was found and with zero offset 18.2% respectively, both observed at 19.5 km (figure 4(d)). Furthermore, the predicted regional background by setting the offset to 18 km (coefficients of the polynomial given in table 1) closely matches with the assumed regional, whereas the other regional fields estimated with 15 and 0 km offsets do not (figure 4(a)).

Field example

The Gediz and Büyük Menderes are two prominent finite strike (150 km) east–west trending onshore grabens in the western Turkey (Eyidogan and Jackson 1985, Paton 1992, Sari and Şalk 2002). These grabens, bounded by normal fault systems, are filled with thick-sectioned sediments above the metamorphic basement complex. The master fault bounding the Büyük Menderes graben is on the northern side, whereas the master fault associated with the Gediz graben is towards the south (figures 5 and 6).

Paton (1992) had analyzed the gravity anomalies of the two grabens along four selected transects; three across the Gediz graben and one over the Büyük Menderes graben respectively. These interpretations were carried out with an assumption that the sedimentary fill within the grabens is uniform, which however is not so in reality (Sari and Şalk 2002). Based on the borehole data, Sari and Şalk (2002) described the density contrast variation of the sedimentary rocks within the grabens by hyperbolic functions (figure 7) and used them in the analyses of two gravity profiles across the Gediz graben (figures 8 and 9) and one over the Büyük Menderes graben (figure 10). However, the interpretations carried out by both Paton (1992) and Sari and Şalk (2002) are based on 2D approaches. While justifying the need to approximate the grabens to 2.5D sources, Chakravarthi and Sundararajan (2007) have analyzed the gravity anomalies of the Gediz graben for its basement structure along a profile presuming the sediment fill above the basement as collage of vertical prisms (figure 9(b)).

Table 2. Derived hyperbolic (Sari and Şalk 2002) and exponential density contrast models, Gediz and Büyük Menderes grabens, western Turkey.

	Hyperbolic density contrast model		Exponential density contrast model (EDCM)	
Name of the graben/Profile No.	$\Delta\rho_0$ (gm cm^{-3})	β (km)	$\Delta\rho_0$ (gm cm^{-3})	λ (km^{-1})
Gediz/2	−1.407	0.859	−1.182	1.019
Gediz/3	−1.407	0.620	−1.320	1.6708
Büyük Menderes/XY	−0.98	2.597	−0.798	0.3808

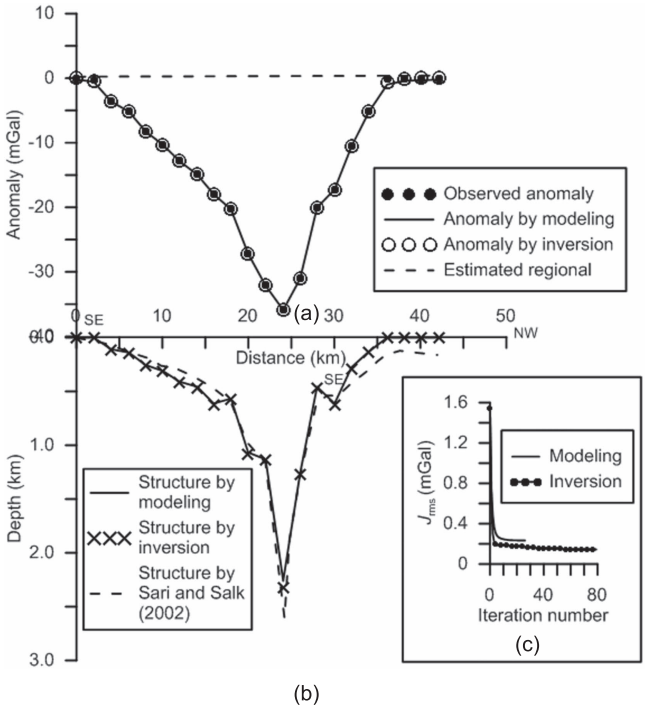


Figure 10. (a) Observed and theoretical gravity anomalies by automatic modeling and inversion along Profile XY, Büyük Menderes graben, western Turkey. Estimated regional is also shown, (b) inferred depth structures of the graben by automatic modeling and inversion. Estimated structure by Sari and Şalk (2002) based on 2D modeling is also shown for comparison, (c) changes in rms errors with iteration.

Table 3. Rms errors for initial and estimated models by automatic modeling and inversion, Gediz and Büyük Menderes grabens, western Turkey.

	Rms error (mGal)			
	Automatic modeling		Inversion	
Name of the graben/ Profile No.	Initial	Final	Initial	Final
Gediz/2	1.59	0.95	1.59	0.07
Gediz/3	0.67	0.02	0.67	0.04
Büyük Menderes/XY	1.54	0.23	1.54	0.14

Table 4. Estimated coefficients of linear regional background, Gediz and Büyük Menderes grabens, western Turkey.

Name of the graben/Profile No.	f_0	f_1
Gediz/2	−4.02	0.110
Gediz/3	0.06	−0.002
Büyük Menderes/XY	0.23	0.004

In the present case, the gravity profiles considered by Sari and Şalk (2002) are re-interpreted by the proposed algorithms using the derived EDCMs of respective grabens. The constants of EDCMs ($\Delta\rho_0$ and λ) obtained by fitting equation (2) to the hyperbolic density contrast models (Sari and Şalk 2002) are given in table 2 and shown in figure 7.

The interpreted results of the three gravity profiles by the proposed techniques are shown in figures 8–10. It is to note that these profiles do not bisect the strike lengths of respective grabens but run at different offsets, hence the offsets of the profiles measured from the geologic maps (Sari and Şalk 2002) are used in the interpretation for reliable results.

The modeling technique has performed 23 iterations for a proper convergence of the anomalies along the Profile 2 and 26 iterations in each case for the Profiles 3 and XY respectively. For the same set of gravity anomalies when inversion has applied it took 6 and 4 iterations for the Profiles 2 and 3 of the Gediz graben, and 79 iterations for the Profile XY of the Büyük Menderes graben, respectively. The modeling and inversion techniques got terminated at the end of the said concluding iterations because one of the termination criteria in each case was fulfilled.

The magnitudes of the initial and final rms errors (equation (11)) between the observed and modeled gravity anomalies in each case are given in table 3 and the changes in rms errors with iteration are shown graphically in figures 8(c), 9(c) and 10(c).

The gravity anomalies subsequent to the analysis are shown in figures 8(a)–10(a) along with the observed ones. In all cases, the nature of fit between the observed and model gravity anomalies is satisfactory (figures 8(a)–10(a)). The coefficients of regional gravity background estimated from the inversion for each one of the profile are given in table 4 and shown in respective figures from 8(a) to 10(a).

The recovered basement structures along the three profiles from automatic modeling comply well with the structures deciphered from inversion (figures 8(b)–10(b)). The maximum depths to the basement estimated from the present analyses and the ones reported by Sari and Şalk (2002) along the three selected profiles under consideration are given in table 5.

By and large, the deciphered structural models of the two grabens from the present analyses (figures 8(b), 9(b) and

Table 5. Maximum depths to the basement estimated from gravity modeling and inversion, Gediz and Büyük Menderes grabens, western Turkey.

Name of the graben/ Profile No.	Distance (km)	Estimated depth from modeling (km)	Estimated depth from inversion (km)	Estimated depth by Sari and Şalk (2002) (km)
Gediz/2	16.0	1.91	1.75	2.07
Gediz/3	46.0	1.65	1.60	1.75
Büyük Menderes/XY	24.0	2.26	2.32	2.60

10(b)) compare reasonably well with those reported by Sari and Şalk (2002), however, with a few exceptions. For e.g., all the structural models derived from the present analyses (from both modeling and inversion) divulge (i) relatively deeper basement over the shoulders (except for the Profile XY on the northwestern part) and (ii) relatively shallower basement in the depocentres compared to the ones reported by Sari and Şalk (2002) (table 5 and figures 8(b)–10(b)). The interpreted depth structure of the Gediz graben along the Profile 3 by the stacked prism model (Chakravarthi and Sundararajan 2007) is also shown in figure 9(b) for comparison. A maximum depth of 1.65 km to the basement estimated at the 46th km on the Profile 3 from the present interpretation remarkably coincides with a figure of 1.64 km inferred by Chakravarthi and Sundararajan (2007).

Discussion

The applicability of the proposed automatic modeling and inversion is exemplified with a synthetic example and also on a few real field gravity anomalies. In case of the synthetic example, pseudorandom noise added to the gravity anomalies produced by a residual source (sedimentary basin) at 18 km offset (treated as observed anomalies) are analyzed by both automatic modeling and inversion to recover the structure; setting the offset parameter to 0, 15 and 18 km independently. At the end of the concluding iteration the theoretical gravity anomalies in each case equally fit the observed noisy anomalies but the estimated depth structures differ from each other. The structures deciphered from both automatic modeling and inversion with 18 km offset are exactly the same and explained well the assumed structure even in the presence of pseudorandom noise. On the other hand, even though the estimated structures from automatic modeling correlate well with the corresponding structures obtained from inversion for different offsets (0 km and 15 km); these structures do not comply well with the assumed one.

In the presence of both regional background and pseudorandom noise the inversion technique by setting 18 km offset has yielded a structure that is more or less consistent with the assumed structure with a few marginal errors. However, the inversions performed on the gravity anomalies with 15 and 0 km offsets fail to recover the structure. In a nutshell, it is demonstrated that in both automatic modeling and inversion the final solution was offset dependent as the further the offset deviates from the optimum offset the more the final solution deviates.

The derived exponential density contrast models for Gediz and Büyük Menderes grabens in western Turkey are used to analyze the observed gravity anomalies of respective grabens. The estimated structures of the two grabens from present modeling and inversion show moderate deviations from the structures previously reported by Sari and Şalk (2002) which are based on the interpretation of gravity anomalies using a 2D approach with hyperbolic density functions.

Conclusion

Two interpretation techniques, based on the principles of automatic modeling and inversion, using the exponential density contrast model are developed in the spatial domain to trace the basement structures of finite strike sedimentary basins from the observed gravity anomalies. The sediment-basement interface is described by a finite-strike polygonal source with several vertices, whose depth ordinates become the unknown parameters to be estimated. Unlike the case with modeling, the inversion technique simultaneously estimates regional trend described by a linear equation. The proposed interpretation techniques are automatic as they generate the initial parameters from the observed gravity anomalies, estimates the improvements in corresponding parameters and modifies them in an iterative approach following the criteria of minimization of rms error between the observed and modeled gravity anomalies. Forward modeling is realized through an approach that combines both analytical and numerical methods, because no closed form analytical gravity expression could be derivable in the spatial domain using an exponential density contrast model. The reliability of the proposed forward modeling is established by comparing the anomalies realized from the present method with those anomalies obtained from an analytical method. It is demonstrated in detail with a synthetic example that the magnitude of gravity anomalies over a finite strike geologic structure is offset dependent and that this parameter plays a vital role in the interpretation.

The advantage of the proposed techniques is that they are fairly applicable to analyze the gravity anomalies even when the profile fails to bisect the strike length of a sedimentary basin.

In the proposed interpretation methodologies it is presumed that the density of underlying basement below the sedimentary column is uniform, and that the regional background along a profile varies linearly as a function of observer location across the strike in case of inversion. However, these assumptions may or may not be valid elsewhere. Therefore, the strategies presented here yield reliable interpretations where the enlisted assumptions are more or less valid.

References

Athy L F 1930 Density, porosity and compaction of sedimentary rocks *Bull. Am. Astron. Pet. Geol.* **14** 1–24

Azab A and El-Khadragy A A 2013 2.5D gravity/magnetic model studies in Sahl El Qaa area, Southwestern Sinai, Egypt *Pure Appl. Geophys.* **170** 2207–29

Backus G E and Gilbert F J 1967 Numerical applications of a formalism for geophysical inverse problems *Geophys. J. R. Astron. Soc.* **13** 247–76

Backus G E and Gilbert F J 1968 The resolving power of gross earth data *Geophys. J. R. Astron. Soc.* **16** 169–205

Backus G E and Gilbert F J 1970 Uniqueness in the inversion of inaccurate gross earth data *Phil. Trans. R. Soc.* A **226** 123–92

Bhaskara Rao D 1986 Modeling of sedimentary basins from gravity anomalies with variable density contrast *Geophys. J. R. Astron. Soc.* **84** 207–12

Bhaskara Rao D 1990 Analysis of gravity anomalies of sedimentary basins by an asymmetrical trapezoidal model with quadratic density function *Geophysics* **55** 226–31

Bhaskara Rao D and Mohan Rao C P V N J 1999 Two-dimensional interpretation of gravity anomalies over sedimentary basins with an exponential decrease of density contrast with depth *Proc. Indian Academy of Sciences-Earth and Planetary Sciences* vol 108, pp 99–106

Bhaskara Rao D, Prakash M J and Ramesh Babu N 1993 Gravity interpretation using Fourier transforms and simple geometrical models with exponential density contrast *Geophysics* **58** 1074–83

Bott M H P 1960 The use of rapid digital computing methods for direct gravity interpretation of sedimentary basins *Geophys. J. R. Astron. Soc.* **3** 63–7

Braitenberg C, Wienecke S and Wang Y 2006 Basement structures from satellite-derived gravity field: South China Sea ridge *J. Geophys. Res.—Solid Earth* **111** B05407

Cai H and 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** G81–94

Castagna J P, Batzle M L and Kan T K 1993 Rock physics—the link between rock properties and AVO response *Offset-Dependent Reflectivity—Theory and Practice of AVO Analysis* (*Investigations in Geophysics Series* 8) ed P Castagna and M M Backus (Tulsa, OK: Society of Exploration Geophysicists) pp 124–57

Chai Y and Hinze W J 1988 Gravity inversion of an interface above which the density contrast varies exponentially with depth *Geophysics* **53** 837–45

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 2009 Automatic gravity inversion for simultaneous estimation of model parameters and regional gravity background: an application to 2D pull-apart basins *Curr. Sci.* **96** 1349–60

Chakravarthi V, Pramod Kumar M, Ramamma B and Rajeswara Sastry S 2016 Automatic gravity modeling of sedimentary basins by means of polygonal source geometry and exponential density contrast variation: two space domain based algorithms *J. Appl. Geophys.* **124** 54–61

Chakravarthi V, Rajeswara Sastry S and Ramamma B 2013 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 *Comput. Geosci.* **56** 131–41

Chakravarthi V and Sundararajan N 2004 Ridge regression algorithm for gravity inversion of fault structures with variable density *Geophysics* **69** 1394–404

Chakravarthi V and Sundararajan N 2006 Discussion on the gravitational attraction of a right rectangular prism with density varying with depth following a cubic polynomial *Geophysics* **71** X17–9

Chakravarthi V and Sundararajan N 2007 3D gravity inversion of basement relief—a depth dependent density approach *Geophysics* **72** I23–32

Chappell A and Kusznir N 2008 An algorithm to calculate the gravity anomaly of sedimentary basins with exponential density-depth relationships *Geophys. Prospect.* **56** 249–58

Cordell L 1973 Gravity anomalies using an exponential density-depth function-San Jacinto Graben, California *Geophysics* **38** 684–90

Cowie P A and Karner G D 1990 Gravity effect of sediment compaction: examples from the North Sea and Rhine Graben *Earth Planet. Sci. Lett.* **99** 141–53

D'Urso M G 2014 Gravity effects of polyhedral bodies with linearly varying density *Celest. Mech. Dyn. Astron.* **120** 349–72

Eyidogan H and Jackson J A 1985 A seismological study of normal faulting in the Demirci, Alasehir and Gediz earthquakes of 1969–1970 in W Turkey: implications for the nature and geometry of deformation in the continental crust *Geophys. J. R. Astron. Soc.* **81** 569–607

Filon L N G 1928 On a quadrature formula for trigonometric integrals *Proc. R. Soc. Edin.* **49** 38–47

Gallardo-Delgado L A, Perez-Flores M A and Gomez-Trevino E 2003 A versatile algorithm for joint inversion of gravity and magnetic data *Geophysics* **68** 949–59

Garcia-Abdeslem J 2005 The gravitational attraction of a right rectangular prism with density varying with depth following a cubic polynomial *Geophysics* **70** j39–42

Granser H 1987 Three-dimensional interpretation of gravity data from sedimentary basins using an exponential density-depth function *Geophys. Prospect.* **35** 1030–41

Hamayun K, Prutkin I and Tenzer R 2009 The optimum expression for the gravitational potential of polyhedral bodies having a linearly varying density distribution *J. Geod.* **83** 1163–70

Hansen R O 1999 An analytical expression for the gravity field of a polyhedral body with linearly varying density *Geophysics* **64** 75–7

Kinsmann D J J 1975 Rift basins and peculiarities of sediment accumulation in the conditions of sagged margins of continents *Petroleum and Global Tectonics* ed G Fischer and S Judson (Princeton, NJ: Princeton University Press) pp 61–91

Klinger F L, Nacif S, Martinez M P, Gimenez M E, Ruiz F and Alvarez O 2011 Gravimetric model of the Gastre trough, province of Chubut *Argentina: Boletín Geológico y Minero* **122** 299–310

Manger G E 1963 Porosity and bulk density of sedimentary rocks, Contributions to Geochemistry, U.S. Atomic Energy Commission *US Geological Survey Bulletin* 1144-E E1–56

Marquardt D W 1970 Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation *Technometrics* **12** 591–612

Martelet G, Perrin J, Truffert C and Deparis J 2013 Fast mapping of magnetic basement depth, structure and nature using aeromagnetic and gravity data: combined methods and their application in the Paris Basin *Geophys. Prospect.* **61** 857–73

Maxant J 1980 Variation of density with rock type, depth, and formation in the Western Canada basin from density logs *Geophysics* **45** 1061–76

Mickus K L and Peeples W J 1992 Inversion of gravity and magnetic data for the lower surface of a two and one-half dimensional sedimentary basin *Geophys. Prospect.* **40** 171–93

Paton S 1992 Active normal faulting, drainage patterns and sedimentation in southwestern Turkey *J. Geol. Soc.* **149** 1031–44

Pedersen L B 1985 The gravity and magnetic fields from ellipsoidal bodies in the wave number domain *Geophys. Prospect.* **33** 263–81

Radhakrishna Murthy I V 1998 *Gravity and Magnetic Interpretation in Exploration Geophysics* (*Memoir* 40) (Bengaluru: Geological Society of India)

Rama Rao P and Radhakrishna Murthy I V 1989 Two fortran77 function subprograms to calculate gravity anomalies of bodies of finite and infinite strike length with the density contrast differing with depth *Comput. Geosci.* **15** 1265–77

Ramillien G and Wright I C 2002 Sea mount gravity anomaly modeling with variably thick sediment cover *Mar. Geophys. Res.* **23** 13–23

Reamer S K and Ferguson J F 1989 Regularized two-dimensional Fourier gravity inversion method with application to the Silent Canyon caldera, Nevada *Geophysics* **54** 486–96

Rusakov O M 1990 The thickness and density of sedimentary cover in the indian ocean *Geophys. J.* **3** 390–9

Sari C and Şalk M 2002 Analysis of gravity anomalies with hyperbolic density contrast: an application to the gravity data of Western Anatolia *J. Balkan Geophys. Soc.* **5** 87–96

Tenzer R and Gladkikh V 2014 Assessment of density variations of marine sediments with ocean and sediment depths *Sci. World J.* **2014** 823296

Visweswara Rao C, Chakravarthi V and Raju M L 1994 Forward modelling: gravity anom-alies of two-dimensional bodies of arbitrary shape with hyperbolic and parabolic density functions *Comput. Geosci.* **20** 873–80

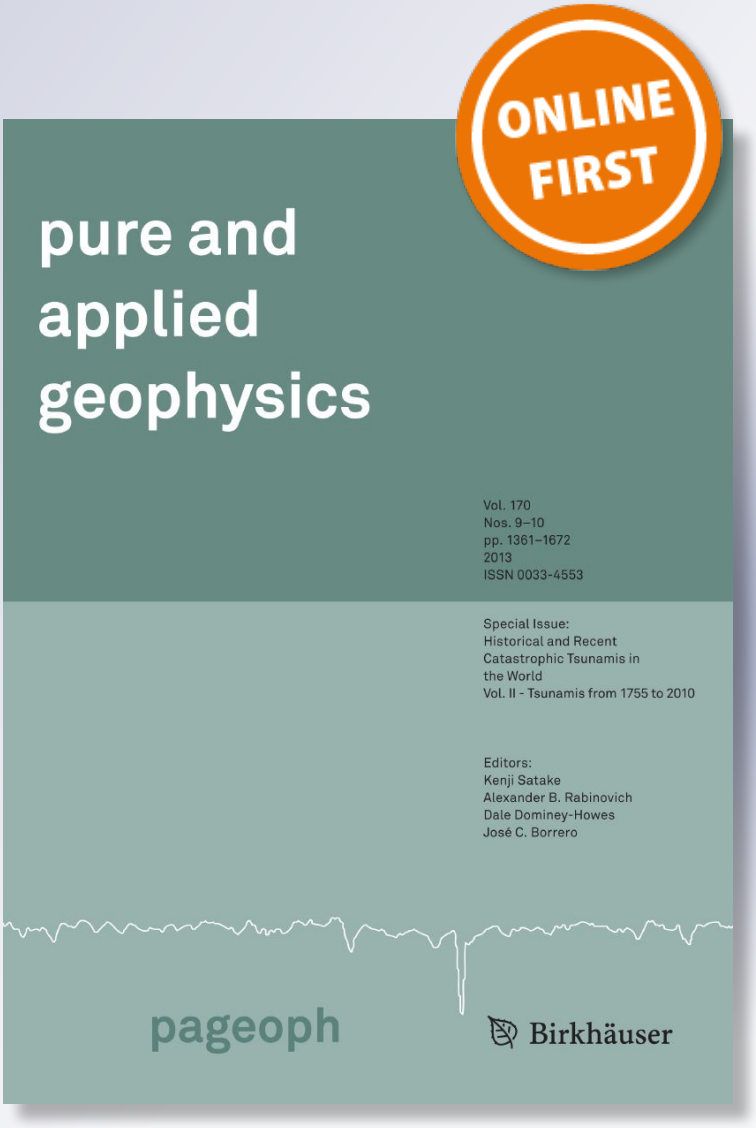
3D Gravity Analysis in the Spatial Domain: Model Simulation by Multiple Polygonal Cross-Sections Coupled with Exponential Density Contrast

K. Mallesh, V. Chakravarthi & B. Ramamma

Pure and Applied Geophysics
pageoph

ISSN 0033-4553

Pure Appl. Geophys.
DOI 10.1007/s00024-019-02103-9



Springer

Pure Appl. Geophys.

https://doi.org/10.1007/s00024-019-02103-9

Pure and Applied Geophysics

3D Gravity Analysis in the Spatial Domain: Model Simulation by Multiple Polygonal Cross-Sections Coupled with Exponential Density Contrast

K. Mallesh,[1] V. Chakravarthi,[1] and B. Ramamma[1]

Abstract—An automatic 3D modeling technique is developed in the spatial domain to analyze the gravity anomalies produced by a concealed density interface with mass density contrast differing exponentially with depth. The sedimentary column above the interface is described with a stack of multiple vertical polygonal sections of unit thickness each. For such a case, the depth ordinates of the vertices of the cross-sections become the unknown parameters to be estimated from gravity data. Forward solution of the model space is realized in the spatial domain by a technique that combines both analytic and numeric approaches. Initial depths to the interface are calculated based on the Bouguer slab approximation and subsequently improved, iteratively, based on the ratio of the product of the observed gravity anomaly and existing depth parameter to the corresponding model gravity response. The iterative process continues until one of the predefined termination criteria is accomplished. Unlike the existing methods, the advantage of the proposed method is that the observed gravity anomalies need not necessarily be sampled/available at regular spatial grid intervals. The applicability of the proposed model is exemplified with a set of noisy gravity anomalies attributable to a synthetic structure before being applied to a real world gravity data. In the case of the synthetic example, the method has yielded a structure that was compatible with the assumed structure even in the presence of random noise. Application of the proposed method to the gravity data set from the Los Angeles Basin, California, using a prescribed exponential density function has yielded a model that concurs reasonably well with the published models.

Key words: Density interfaces, gravity anomalies, 3D automatic modeling, polygonal cross-sections, exponential density model.

1. Introduction

Although the gravity method is one among the age old geophysical methods, it still plays a pivotal role to address a variety of geological problems associated not only with the exploration of natural resources, but also various geodynamic/tectonic processes. In regional and hydrocarbon explorations, the gravity method provides vital information on the disposition and orientation of sedimentary basins, even when the basins are concealed under a thick pile of basaltic cover (Chakravarthi et al. 2007). The information gained from a gravity survey over a region is very informative to plan and execute seismic surveys in a more systematic manner.

Estimating the depths to concealed density interfaces from observed gravity anomalies is an important application of the gravity method. However, due to the inherent ambiguity associated with the potential field methods, gravity data are often analysed in the light of known information supplemented by either drilling or other geophysical data. Three-dimensional interpretation of gravity anomalies is generally carried out in two modes, namely, (1) estimating the subsurface density distribution by discretizing the model space into a large number of cells, where each cell possesses uniform density (Li and Oldenburg 1998; Rezaie et al. 2017); or (2) to estimate the optimum parameters of prescribed geometries (used to simulate subsurface geology) by specifying the density contrast/s as a known parameter. Among these two, the latter approach has gained more importance because the number of unknown parameters to be estimated, as well as the model uncertainty can be reduced significantly (Cai et al. 2018). In either approach, the misfit between the observed and model gravity anomalies shall be minimized following some type of fitting criteria.

Forward modelling, i.e., computation of gravity response of a prescribed model space defined with a known set of shape and size parameters, forms an important exercise in automatic modelling and inversion strategies. Talwani and Ewing (1960) proposed an ingenious method to compute the gravity

[1] Centre for Earth, Ocean and Atmospheric Sciences, University of Hyderabad, Gachibowli, Hyderabad 500046, India. E-mail: malleshkamal08@gmail.com

Published online: 15 February 2019

response of a 3D homogenous source at an external point, wherein the gravity effects of several horizontal polygonal laminas representing the outline of the source volume were computed using a constant density and numerically integrated to obtain the total gravity effect of the source. However, it becomes difficult to use such a method in automatic modelling schemes to model the gravity anomalies resulting from sedimentary basins. This is because the modelling result would be severely affected when the depth ordinates of the lamina vertices are kept unchanged while allowing the horizontal coordinates of the vertices to change during the process of inversion (Rao et al. 1999). On the other hand, Nagy (1966), Banerjee and Das Gupta (1977), and Tsoulis (2000) have presented equations allowing one to compute the gravity anomalies due to parallelepipeds, and for polyhedral bodies, Paul (1974), Okabe (1979), Pohanka (1988), Holstein (2002), and D'Urso (2014a) have provided formulae. Again, all of these schemes presume uniform density for the sources.

Three-dimensional techniques to analyze the gravity anomalies of sedimentary basins, treating the sediment density as uniform throughout its volume, are available (see e.g., Cordell and Henderson 1968; Pilkington and Crossley 1986; Murthy et al. 1990; Comacho et al. 1997; Debeglia and Corpell 1997; Zhdanov and Cai 2013). It is well established that the density of sedimentary rocks is seldom uniform with increasing depth due to the effects of several factors that include stratigraphic layering, facies change, cementation, compaction due to geostatic pressure (Cordell 1973; Papp and Kalmar 1995; Grabowska et al. 1998; Kadirov 2000; Crosby et al. 2006; Tenzer and Gladkikh 2014; Cai and Zhdanov 2015). Therefore, the practical utility of the enlisted 3D modeling schemes fall short of analyzing the gravity anomalies once the sedimentary rocks possess non-uniform density. On the other hand, Hansen (1999), Holstein (2003) and D'Urso (2014b) have presented formulae to calculate the gravity anomalies due to a polyhedral source using linear density function, Chakravarthi et al. (2002) and Abdeslem (2005) have derived analytical gravity expressions due to a right rectangular prism using parabolic and cubic polynomial density functions, and Wu and Chen (2016) have presented a formula using general polynomial functions. Gokula and Sastry (2015) have adopted the parabolic density function to obtain a formula to realize forward modeling of a vertical pyramid that is bounded on the top and bottom by planar surfaces. Nasuti and Ardestani (2007) have proposed a forward modeling scheme to calculate the gravity anomalies of 3D sedimentary basins using a quadratic density function.

Bhaskara Rao et al. (1990), Gallardo-Delgado et al. (2003) and Feng et al. (2016) have presented 3D algorithms to estimate the depths of basement interfaces from the observed gravity anomalies using a quadratic density-depth distribution. Chakravarthi (2003) and Chakravarthi and Sundararajan (2004a) have devised automatic schemes using a parabolic density function to interpret the gravity anomalies resulting from sedimentary basins. Later, Isik and Senel (2009) and Bal and Kara (2012) adopted the 3D interpretation methodologies proposed by Chakravarthi (2003) and Chakravarthi and Sundararajan (2004a) to analyze the gravity anomalies of the Büyük Menderes River Basin and the Salt Lake region of Turkey, respectively. Unambiguously, the linear density function is more appropriate to describe the density variation of sedimentary rocks at larger depths, whereas the quadratic and cubic polynomial density functions poses serious problems in the interpretation particularly when the known density-depth information is available only up to shallower depths of a sedimentary basin (Chakravarthi and Sundararajan 2006; Chakravarthi 2009, 2011). In contrast, Maxant (1980) has reported from the studies of gamma–gamma logs in the Western Canada Sedimentary Basin that the density variation of shale samples with depth could be explained more effectively by an exponential function. Nelson and Fairchild (1989) have demonstrated that the densities of sedimentary rocks from the Gulf of Mexico exhibit rapid increase at shallower depths up to 1.5 km. Cowie and Karner (1990) have illustrated that the mean sediment density derived from several density logs unveils highest rate of increase in the top few kilometres. Based on density samples collected from the Deep Sea Drilling Project, Gu et al. (2014) have shown that high contrast in density exists between the sediments and basement rock where the sediment thickness is minimum, while the density contrast attains its minimum at the lower stratigraphic unit of a thick sedimentary sequence. Such variation of sediment density with depth could be

effectively modeled using an exponential function rather than the prevailing mathematical functions. Therefore, modeling and inversion strategies necessarily warrant the use of the exponential density function to analyze the gravity anomalies of sedimentary basins to obtain more reliable interpretations.

Many of the present-day 3D interpretation techniques that make use of the exponential density function to analyze the gravity anomalies are being developed in the wave number domain because of the simple fact that analytical solutions for forward modeling is underivable in the spatial domain. For this reason, Granser (1987), Chai and Hinze (1988), Feng et al. (2016) and Pham et al. (2018) have performed the exercise of forward computations of gravity anomalies of a 3D source in the frequency domain and transformed the anomalies back to the spatial domain for further analysis. It is obvious that such a procedure of transforming the anomalies from the frequency to the spatial domain invariably experiences truncation errors as the data window is always limited (Chakravarthi and Sundararajan 2004b).

In this paper, we present a technique in the spatial domain to calculate the gravity anomalies of 3D sources in which the density contrast obeys exponential decrease with depth. This technique judiciously combines both analytic and numeric approaches. We also develop an automatic method using the principles of automatic modelling (Chakravarthi et al. 2013) to recover the basement structures from the observed gravity anomalies based on a predefined convergence criterion. We apply this technique to analyse the gravity anomalies produced by a synthetic structure in the presence of pseudorandom noise and then over a real world gravity data pertaining to the Los Angeles Basin, California. In either case, the density contrast varies exponentially with depth.

2. Forward Modeling

In the Cartesian coordinate system, let the z-axis be positive vertically downwards with the x-axis running transverse to it (Fig. 1). The y-axis is perpendicular to the xz plane and directed inwards along which the strike of the model space is scaled (Fig. 1). The gravity anomaly, $\Delta g(0,0,0)$, of this model at a point $O(0, 0, 0)$ can be expressed as:

$$\Delta g(0,0,0) = G \int_v \frac{\Delta\rho(z) z \mathrm{d}v}{(x^2+y^2+z^2)^{3/2}}, \tag{1}$$

where (x, y, z) are the coordinates of an elementary volume dv within the source. The density contrast $\Delta\rho(z)$ of sediments within the source volume at any depth, z, is given by (Cordell 1973; Chakravarthi et al. 2017):

$$\Delta\rho(z) = \Delta\rho_0 \mathrm{e}^{-\lambda z}. \tag{2}$$

Here, $\Delta\rho_0$ is the surface density contrast of the sediments with respect to the undisturbed basement and λ is a decay constant.

Upon describing the model space by a stack of multiple vertical laminas, each one with unit thickness, Eq. (1) can be expressed as:

$$\Delta g(0,0,0) = G \int_{Y1}^{Y2} \left[\int_s \frac{\Delta\rho(z) z \mathrm{d}s}{(x^2+y^2+z^2)^{3/2}} \right] \mathrm{d}y, \tag{3}$$

where, $Y1$ and $Y2$ are the limits of the model space as measured from the point of calculation and the integration is carried out numerically using the trapezoidal rule. Substituting Eq. (2) and applying Stokes' theorem, Eq. (3) becomes:

$$\Delta g(0,0,0) = G\Delta\rho_0 \int_{Y1}^{Y2} \left[\oint_z \left[\frac{xz\mathrm{e}^{-\lambda z}}{(y^2+z^2)(x^2+y^2+z^2)^{1/2}} \right] \mathrm{d}z \right] \mathrm{d}y, \tag{4}$$

To solve the inner integral in Eq. (4), x needs be expressed in terms of z. This could be realized by approximating the outline of a vertical lamina by a multifaceted polygon AA'CDEF…(as shown in Fig. 1). If θ is the angle made by the side CD with the horizontal passing through the vertex C (parallel to x-axis) then

$$x = x_k - z_k \cot\theta + z\cot\theta, \tag{5}$$

where (x_k, z_k) are the co-ordinates of the vertex C. Upon substitution of Eq. (5), Eq. (4) takes the form

$$\Delta g(0,0,0) = \int_{Y1}^{Y2} \Delta g(0,y,0) \mathrm{d}y, \tag{6}$$

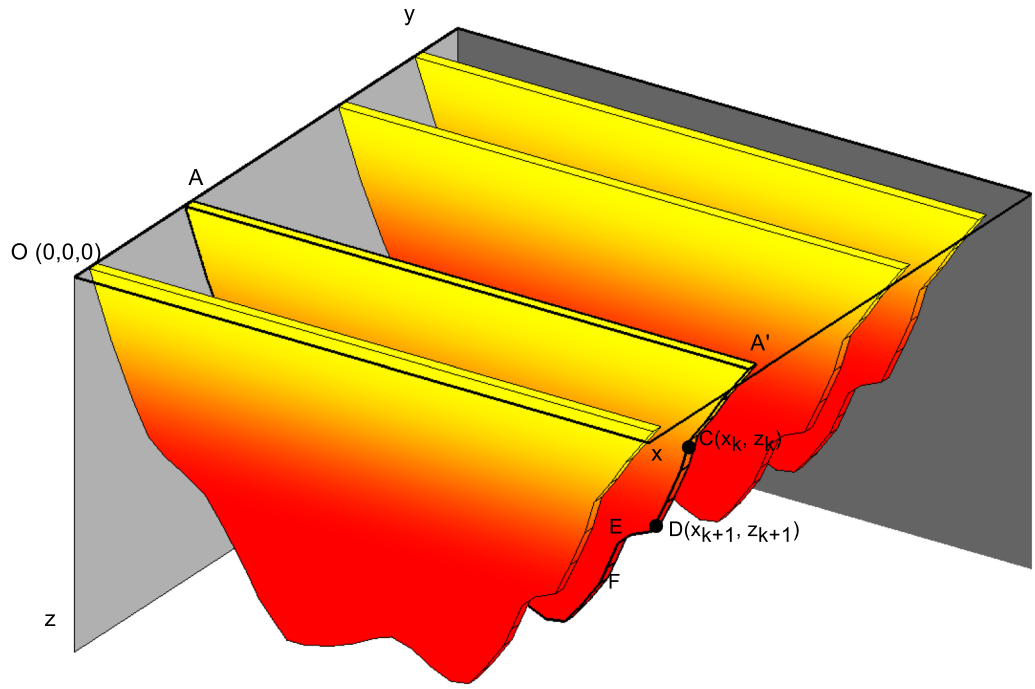


Figure 1
Schematic representation of a 3D sedimentary basin by a stack of vertical polygonal cross-sections in the Cartesian coordinate system

where

$$\Delta g(0,y,0) = G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} \frac{(x_k - z_k \cot\theta + z\cot\theta) z e^{-\lambda z}}{(y^2+z^2)\left[(x_k - z_k\cot\theta + z\cot\theta)^2 + y^2 + z^2\right]^{1/2}} \mathrm{d}z. \tag{7}$$

Here, N is the number of faces bounded by a polygon in the xz-plane. Upon simplification, Eq. (7) can be rewritten as,

$$\Delta g(0,y,0) = G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} \frac{(x_k \sin\theta + \overline{z-z_k}\cos\theta) z e^{-\lambda z}}{(y^2+z^2)\left(\overline{x_k^2+y^2+z^2}\sin^2\theta + \overline{z-z_k}^2\cos^2\theta + x_k\overline{z-z_k}\sin 2\theta\right)^{\frac{1}{2}}} \mathrm{d}z, \tag{8}$$

Here, z_{k+1} is the depth ordinate of the vertex D. Also,

$$\sin\theta = \frac{z_{k+1}-z_k}{\left[(z_{k+1}-z_k)^2 + (x_{k+1}-x_k)^2\right]^{1/2}}, \tag{9}$$

$$\cos\theta = \frac{x_{k+1}-x_k}{\left[(z_{k+1}-z_k)^2 + (x_{k+1}-x_k)^2\right]^{1/2}}, \tag{10}$$

where x_{k+1} is the x coordinate of the vertex D. The coordinates of the vertices are expressed in km and the density contrast in g/cm^3, which results the gravity anomalies measured in mGal. We solve Eq. (8) numerically because no closed form analytic solution could be derivable in the spatial domain. It is of note that the vertices of a polygon, whose coordinates are scaled with reference to the point of calculation, are covered sequentially in the clockwise direction (Chakravarthi et al. 2017) with respect to the increasing y axis. In case the vertices are covered anticlockwise, the magnitude of the anomalous field remains the same, but its sign changes. For a horizontal segment of a polygon, the integral in Eq. (8) takes the form:

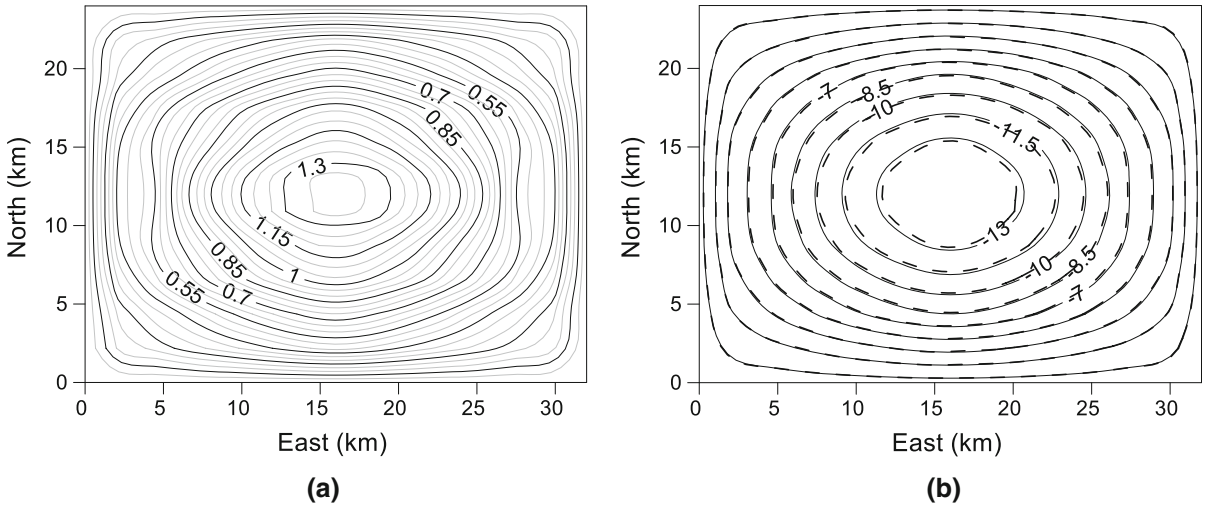


Figure 2
a Geometry of a synthetic homogenous sedimentary basin. Depth contours are drawn at 0.05 km interval, **b** model gravity anomalies obtained from the present method are shown in solid lines whereas the anomalies realized from the method of Talwani and Ewing (1960) are shown in dashed lines. Note that the anomaly contours are drawn at 1.5 mGal interval

$$\int_{z_k}^{z_{k+1}} \frac{z e^{-\lambda z}}{(y^2+z^2)} dz, \tag{11}$$

which ultimately becomes zero. Furthermore, Eq. (8) has a singularity point for an outcropping cross-section at $y = 0$. In such a case the corresponding cross-section is treated as a 2.5D polygon with unit strike length ($\Delta Y = 2Y = 1$) within which the density contrast decreases according to Eq. (2) (Chakravarthi et al. 2017). The gravity anomaly of such a source can be expressed as (Chakravarthi et al. 2017):

$$\Delta g_k = 2G\Delta\rho_0 \sum_{k=1}^{N} \int_{z_k}^{z_{k+1}} e^{-\lambda z} \tan^{-1} \frac{\cos\theta}{2\left(0.25\sin^2\theta + \left(\frac{z}{\Delta Y}\right)^2\right)^{1/2}} dz. \tag{12}$$

The gravity contributions of several representative polygons are calculated and finally integrated over the strike length of the model space to obtain its total gravity anomaly at the point of calculation.

The validity of the proposed forward modeling is examined by comparing the model gravity response of a synthetic homogeneous sedimentary basin (geometry is shown in Fig. 2a) obtained from Eq. (6) with the anomalous field realized through the method of Talwani and Ewing (1960). In this case, the maximum thickness of the sedimentary basin is 1.4 km and the presumed density contrast of the sediments is -0.32 g/cm^3.

The gravity anomalies of the structure calculated at 825 observations using Eq. (6) (shown in Fig. 2b as solid lines) closely resemble the anomalous field obtained from the method of Talwani and Ewing (1960) (represented with dashed lines in Fig. 2b). Hence, the authenticity of the proposed forward modeling scheme is established.

3. Modeling of Gravity Anomalies

Let N_x and N_y be the number of observations along the x-axis and y-axis, respectively. It is presumed that the observed gravity anomalies over the topography are available over and beyond the basin boundary so that the relief of the basin all along its periphery becomes zero. The problem of gravity interpretation then becomes to estimate $(N_x - 2) * (N_y - 2)$ depth parameters from $N_x * N_y$ observed gravity anomalies. Furthermore, known information on the basement depths, if any, available at isolated nodes/observations (e.g., from boreholes) can be specified as constraints in the modeling. To start with, the gravity anomaly at each observation is presumed as being produced by a slab of infinite horizontal

extent below the respective observation. The density contrast within the slab obeys exponential decrease following Eq. (2). The thickness of the slab (z_B) and the gravity anomaly produced by such a slab (g_B) are related to each other by (Cordell 1973; Chakravarthi et al. 2016):

$$z_B = \frac{-1}{\lambda}\log\left(1 - \frac{\lambda g_B}{2\pi G\Delta\rho_0}\right). \tag{13}$$

Equation (9) forms a basis to estimate the initial depths, $z_{(x_i,y_j)}$, of a density interface at any observation, (x_i, y_j), by substituting the corresponding observed gravity anomaly, $\Delta g_{(x_i,y_j)}$, in place of g_B. Subsequently, Eq. (6) calculates the model gravity response of the sedimentary basin, $\Delta g_{M(x_i,y_j)}$, at plurality of observations/nodes, (x_i, y_j), for $i = 1, 2, \ldots, N_x$ and $j = 1, 2, \ldots, N_y$. The difference between the observed $(\Delta g_{(x_i,y_j)})$, and model gravity $(\Delta g_{M(x_i,y_j)})$ anomalies is quantified by r.m.s. error defined by:

$$J = \sqrt{\frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left[\Delta g_{E(x_i,y_j)}\right]^2}{N_x * N_y}}, \tag{14}$$

where

$$\Delta g_{E(x_i,y_j)} = \Delta g_{(x_i,y_j)} - \Delta g_{M(x_i,y_j)}. \tag{15}$$

The depth estimates of the basin are improved, automatically, in an iterative approach based on the equation (Cordell and Henderson 1968; Blakely 1996):

$$z^{k+1}_{(x_i,y_j)} = \frac{\left[\Delta g_{(x_i,y_j)} * z^{k}_{(x_i,y_j)}\right]}{\Delta g^{k}_{M(x_i,y_j)}}. \tag{16}$$

Here, k stands for iteration number and $z^{k+1}_{(x_i,y_j)}$ is the improved depth estimate of the basin at any observation, (x_i, y_j). Eq (12) has the advantage over the existing methods of improving the depth estimates (Cordell 1973; Granser 1987; Chakravarthi and Sundararajan 2004a; Feng et al. 2016) in the sense that it does not contain the density contrast term in the denominator of the right-hand term. In the above methods, when the magnitude of the density contrast becomes too small (at basement depths) there is always a possibility that the depth improvements calculated through the repeated use of the Bouguer formula may lead to overshoot the depth improvements.

The process of updating depth estimates continues till the specified number of iterations completed or the misfit in Eq. (14) falls below a predefined allowable error or the magnitude of the current misfit exceeds its previous value.

4. Examples

The method of interpretation described above is applied to analyze the gravity anomalies on both synthetic and real-world examples.

4.1. Synthetic Example

Figure 3a shows in plan view the geometry of a typical intracratonic sedimentary basin with some type of east–west sagging at the centre. The basement shows progressively increasing dips towards the basin's depocentre with distinct sharp gradients beyond the depths of 0.6 km and 1.6 km, respectively (Fig. 3a). The maximum depth to the floor of the basin (2.93 km) is confined between $x \in$ (14.35 km, 17.92 km) and $y \in$ (10.62 km, 13.37 km). Further, the density contrast of the sediments within the basin is varying exponentially with depth defined by the constants $\Delta\rho_0 = -0.32$ g/cm^3 and $\lambda = 0.3$ g/cm^3/km (Eq. 2). The structure anomaly realized through Eq (6) is shown in Fig. 3b. One can clearly notice from Fig. 3a, b that the faulted basement at large depths, characterized by steep gradients, produce hardly any indicative signatures on the anomaly. Treating the structure anomaly as the observed one, the proposed algorithm is applied over the anomalies to estimate the basement depths for which it took 16 iterations before it was terminated. The modeled anomaly at the end of the 16th iteration, the concluding one, exactly matches with the observed anomaly and the corresponding estimated structure precisely coincides with the assumed structure (this case is not shown for brevity).

We also have analyzed the structure anomaly in the presence of pseudorandom noise. In this case, pseudorandom anomaly noise is imposed with zero mean and standard deviation of 0.66 mGal. Considering the noisy anomaly (Fig. 3c) as the observed one, we applied the proposed method to examine

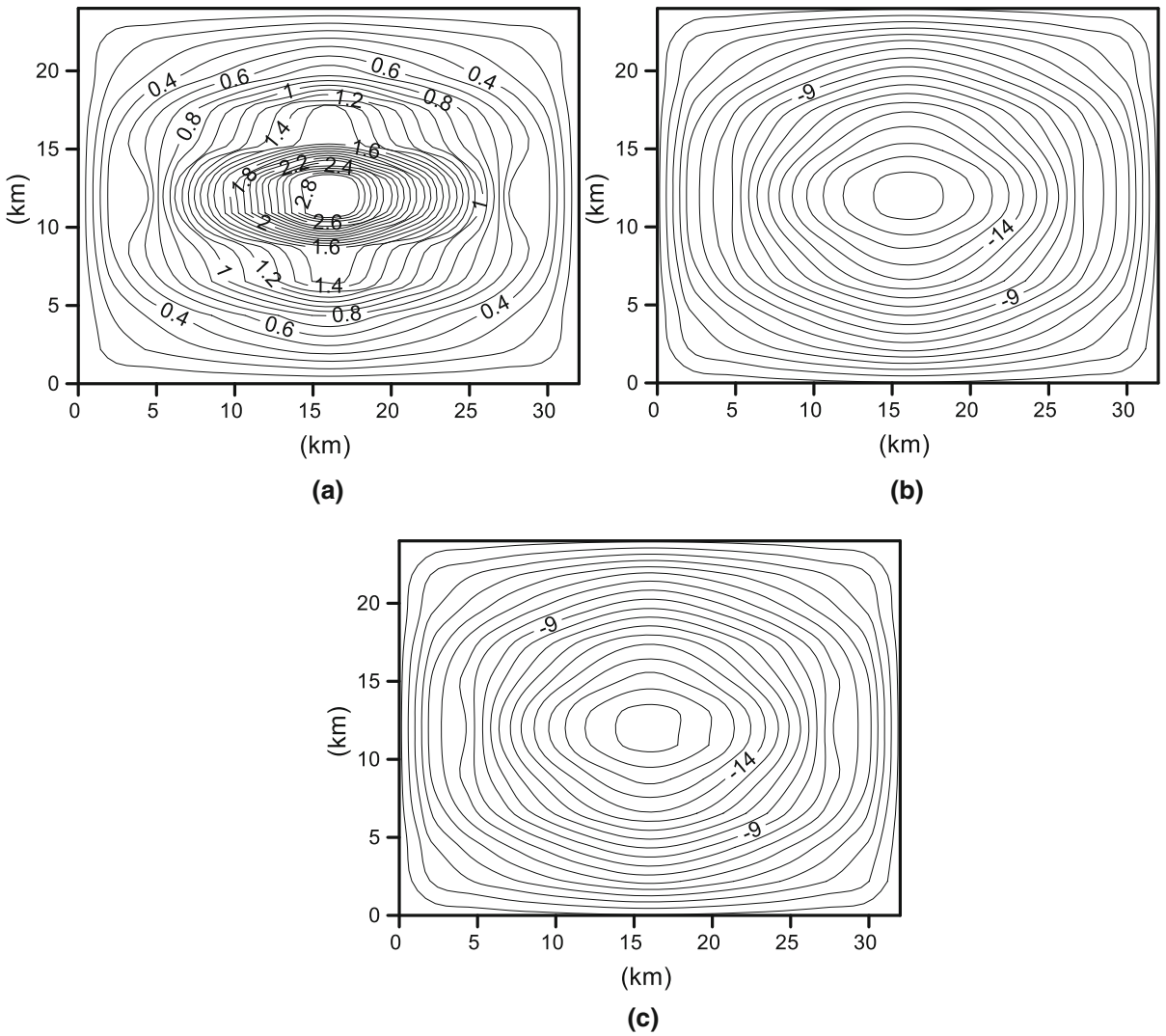


Figure 3
a Plan view of a 3D synthetic model of a sedimentary basin. Depth contours are drawn at 0.1 km interval, **b** gravity response of the basin with exponential density contrast model, **c** gravity response of the structure in the presence of pseudorandom noise. Anomaly contours are drawn at 1 mGal interval

whether the algorithm could recover the structure or not. In this case, the algorithm has performed 20 iterations, beyond which the resulting misfit fell below the predefined error (0.001 mGal), thereby leading to its termination. The initial misfit of 0.9 mGal for the starting model was drastically reduced to 0.048 mGal at the end of the 8th iteration (Fig. 4). The exponential decay in the misfit within the first few iterations is a testimony to rapid convergence of the solution. The interpreted anomaly and the corresponding estimated structure are shown in Fig. 5a, b, respectively. The difference between the observed noisy anomaly and modeled anomaly (after the 20th iteration) is shown in Fig. 5c, whereas, the differences in the assumed and estimated depths are shown in Fig. 5d, respectively. By and large, the modeled anomaly (Fig. 5a) closely resembles the observed anomaly (Fig. 3c), although a few trivial deviations exist in the interpreted anomaly near the depocentre (Fig. 5c). The residuals between the

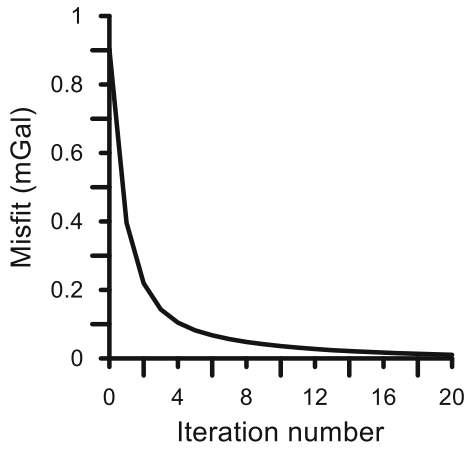


Figure 4
Convergence of the model misfit (mGal) with iteration, synthetic model

observed and modeled anomalies are within the range of ± 0.14 mGal (Fig. 5c). Although, the structural features of the dipping basement were successfully recovered by the algorithm (Figs. 3a, 5b), the predicted depths show marginal deviations from the assumed one particularly near the depocentre (Figs. 3a, 5b, d). For example, the predicted depths to the basement are slightly overestimated by about 2.5% to the north and south of the depocentre, whereas at the centre they are underestimated by 3%. Also, the dimensions of the basement zone at the depocentre were underestimated by 19% along the east–west and 8% along north–south (Figs. 3a, 5b). However, these minor deviations in the estimated structure are within tolerable limits considering the fact that the anomalies used in the modeling are noisy.

4.2. *Field Example*

The Los Angeles Basin in California is known for its great structural relief, complex geologic setting and its hydrocarbon potential. The basin was evolved in five different phases, with each phase represented by distinct rock assemblage (Yerkes et al. 1965). Based on the structure and rock types, the basin is distinguished into four major blocks, namely, southwestern, northwestern, central and northeastern (Fig. 6). Further, each block is delimited by either a major zone of faulting or of flexure in the basement rock (Yerkes et al. 1965). The noteworthy feature of the central block is the presence of a northwest-trending, doubly plunging synclinal trough containing thick-sectioned sediments. The major structural features (Yerkes et al. 1965) and residual gravity anomalies (Chai and Hinze 1988) draped over the digital elevation model (90 m resolution) of the basin (CGIAR-CSI GeoPortal at http://srtm.csi.cgiar.org/) are shown in Fig. 6. One can clearly notice from Fig. 6 that all the structural features of the basin are well highlighted by the anomaly map. For example, the boundary faults of the central block are well correlated with the steep gradients in the gravity anomaly. Similarly, the doubly plunging synclinal trough is manifested by a well-defined negative anomaly. The doming-up of contours in the southwestern block and down-warping of contours in the western part of the central block are well correlated with the mapped anticlinal structures.

Chai and Hinze (1988) have analyzed the gravity anomalies of the basin (Figs. 6, 7a) for its basement structure by assigning a prescribed exponential density contrast to the basin fill defined by $\Delta\rho(z) = -0.5e^{-0.1609z}$, which was obtained from McCulloh's (1960) drill-hole sample density data. For the present case, we digitize the gravity anomaly map (Chai and Hinze 1988) into 375 nodes, sampled at an interval of 3.16 km along the x-axis and 2.82 km along the y-axis (Fig. 7a). The data was interpreted by specifying the allowable misfit between the observed and modelled anomalies as 0.01 mGal. The algorithm performed 157 iterations, after which the resulting misfit has attained a value larger than its previous one, thereby forcing the algorithm to terminate. Overall, the modelled anomaly after the 157th iteration shown in Fig. 7a as a dashed line closely resembles the observed anomaly (solid line in Fig. 7a). A few minor deviations are observed between the two anomalies at the centre of the basin and on the southern boundary.

Figure 7b shows the residuals between the observed and modelled anomalies after the 157th iteration, the concluding one. One can notice that the majority of the misfits across the model space is near zero (Fig. 7b). The estimated structure of the basin is shown in Fig. 8a. The major structural features reported previously by Yerkes et al. (1965) are also

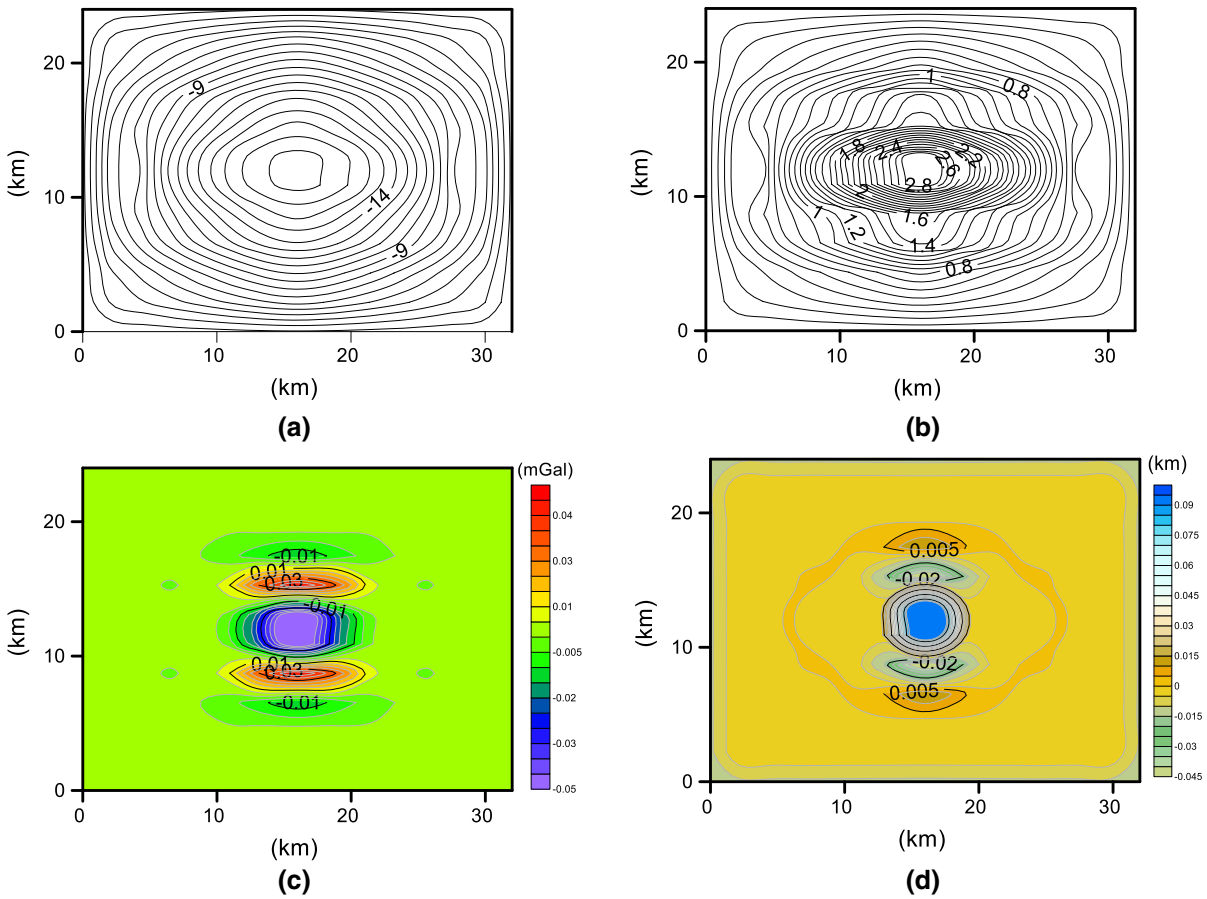


Figure 5

a Modeled gravity anomaly represented with 1 mGal contour interval, **b** estimated structure from the present method. Depth contours are drawn at 0.1 km interval, **c** residuals between the observed and modeled gravity anomalies drawn at 5E−3 mGal contour interval, **d** differences between the assumed and estimated depths at 5E−3 km contour interval, synthetic example

marked on the estimated structure for comparison (Fig. 8a).

One can notice from Fig. 8a that all the structural features reported by Yerkes et al. (1965) are well reflected on the estimated structure too. The inferred structure of the basin by Chai and Hinze (1988) is shown in Fig. 8b for comparison. In this case, the misfit, which has attained its maximum value of 5.08 mGal for the starting model has reduced drastically to 0.45 mGal at the end of the 14th iteration, beyond which the decay was rather gradual (Fig. 9a).

The maximum depth to the floor of the basin obtained from the present method is 9.73 km (Fig. 8a), whereas Chai and Hinze (Fig. 8b) have reported 10 km. The structure contour map prepared by McCulloh (1960) on the basis of surface geology, drill-hole and seismic reflection data has also revealed a figure of 9.5 km for the maximum thickness of sediments. By and large, the modeled structure deciphered from the present method is comparable with the one reported by Chai and Hinze (1988), however, a few exceptions exist. For example, the estimated structure from the present method has revealed the presence of two north–south trending basement depressions (confined between $x \in$ (25 km, 45 km) and $y \in$ (20 km, 35 km)) separated by a basement high. Such a moat-like structure did not reflect prominently in the Chai and Hinze (1988) model. Figure 9b compares the cross-sections of the basin along a selected profile, MM', obtained from the two interpreted models (Fig. 8a, b). Both models have divulged the presence of steeply dipping fault

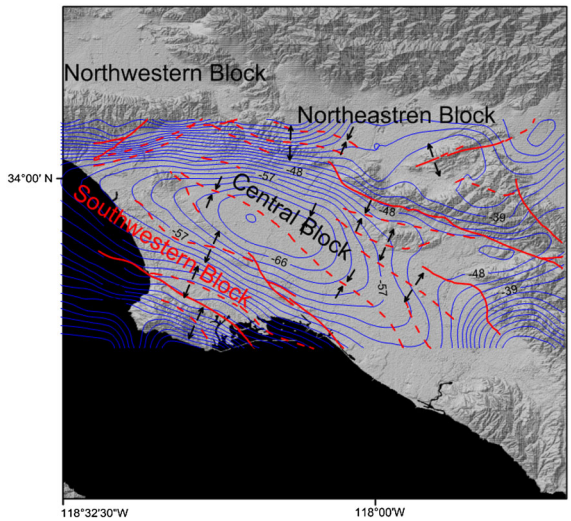


Figure 6
Observed gravity anomalies (after Chai and Hinze 1988), mapped structural features (after Yerkes et al. 1965) and digital elevation model of the Los Angeles Basin, California. Solid lines in blue are the observed gravity anomalies, solid lines in red represent faults or fault zones (dashed where approximately located), solid lines in red with opposite arrow heads indicate anticlines (dashed where approximately located), solid lines in red with arrow heads inwards indicate syncline (dashed where approximately located)

system on the southwestern side beyond a depth of 4 km, whereas on the northeastern side the basin is bounded by a listric fault. Unlike the interpreted model by Chai and Hinze (1988), the present interpreted model indicates that the sedimentary basin appears to extend farther south beyond the boundary of the interpreted grid, which is also well reflected in the structural map of Yerkes et al. (1965).

5. Discussion

Two examples, one synthetic and one real, are presented to demonstrate the applicability of the technique. In the case of the synthetic example, the gravity anomalies produced by a known but complex structure in the presence of pseudorandom noise are analysed. In this case, the algorithm has yielded a structure that is, on the whole, congruent to the assumed structure even in the presence of pseudo-random noise. However, minor deviations are noticed between the assumed and estimated depths in and around the depocentre, which are insignificant considering the fact that the anomalies used in the inversion are noisy.

In real field example, the anomalies resulted from the Los Angeles Basin, California, are analysed. Chai and Hinze (1988) have originally interpreted the anomalies of the basin making use of a derived exponential density contrast model. In the present study, we also have adopted the same density-depth model to analyse the anomalies using our method. The estimated depths of the basin are in close agreement with those inferred by Chai and Hinze (1988) and Yerkes et al. (1965). A few deviations noticed between the present estimated structure and the one by Chai and Hinze (1988) are explained.

6. Conclusion

A space domain based algorithm is presented, to infer the 3D sedimentary basin structure from observed gravity anomalies over the basin, where the density contrast varies exponentially with depth. In contrast to the method of Talwani and Ewing (1960); the present technique approximates the 3D anomalous source by an ensemble of multiple vertical cross-sections of polygonal shape, each one with unit thickness, and the same prescribed density contrast function varying exponentially with depth. Such a simulation of model geometry has the advantage over the method of Talwani and Ewing (1960) in the sense that it is relatively easy to design interpretational strategies to model subsurface geologic structures from the gravity anomalies. Because no closed from analytical solutions could be derivable in the spatial domain using the exponential density function, a method that combines both analytic and numeric approaches is presented to realize forward modelling. The proposed technique calculates the initial depths of a basement structure from the measured gravity anomalies and afterwards improves the depth estimates in an iterative approach till one of the prescribed termination criteria satisfies.

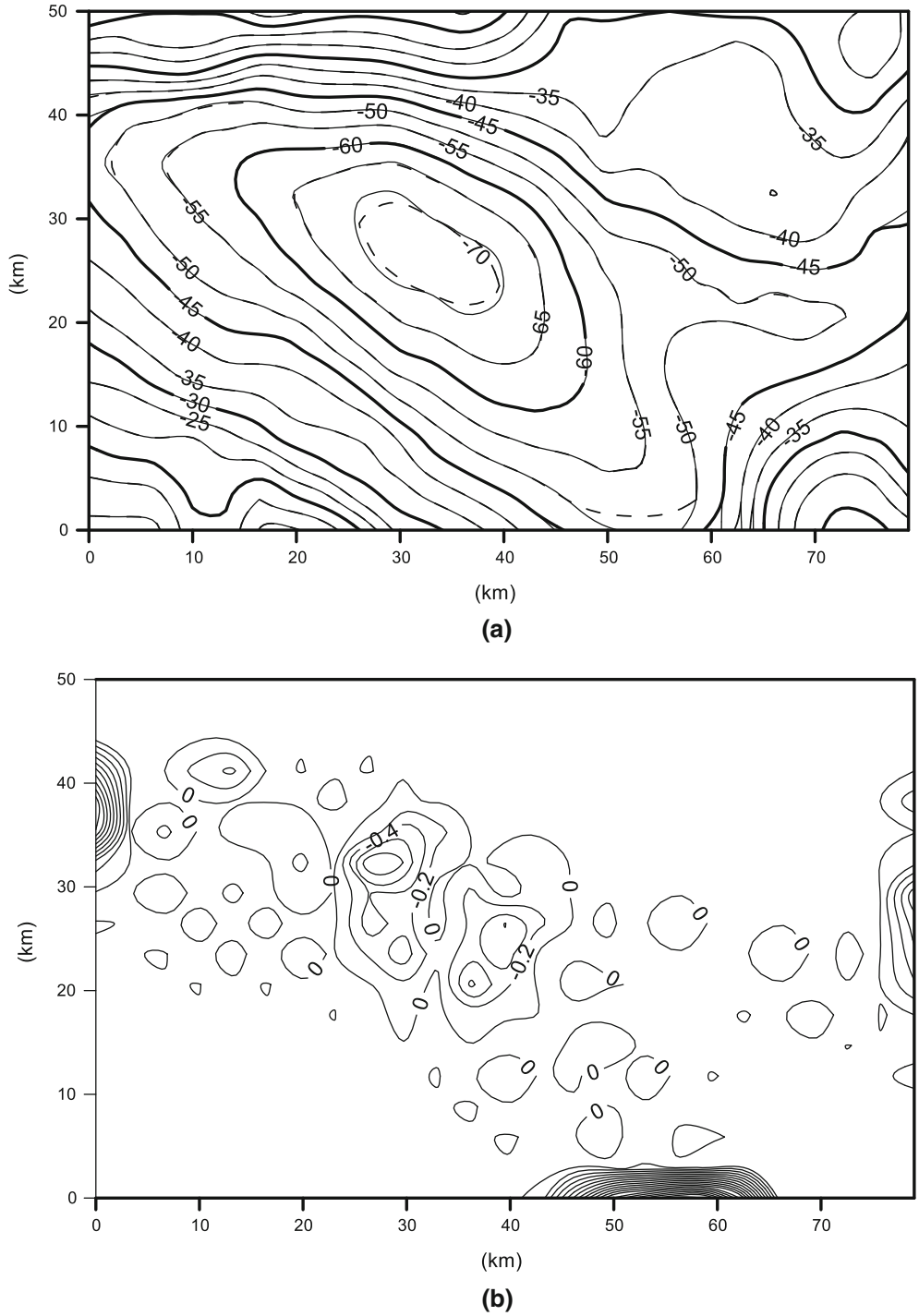


Figure 7
a Observed (solid lines) and modeled gravity anomalies (dashed lines) drawn at 5 mGal contour interval, **b** residuals between the observed and modeled gravity anomalies drawn at 0.2 mGal contour interval, Los Angeles Basin, California

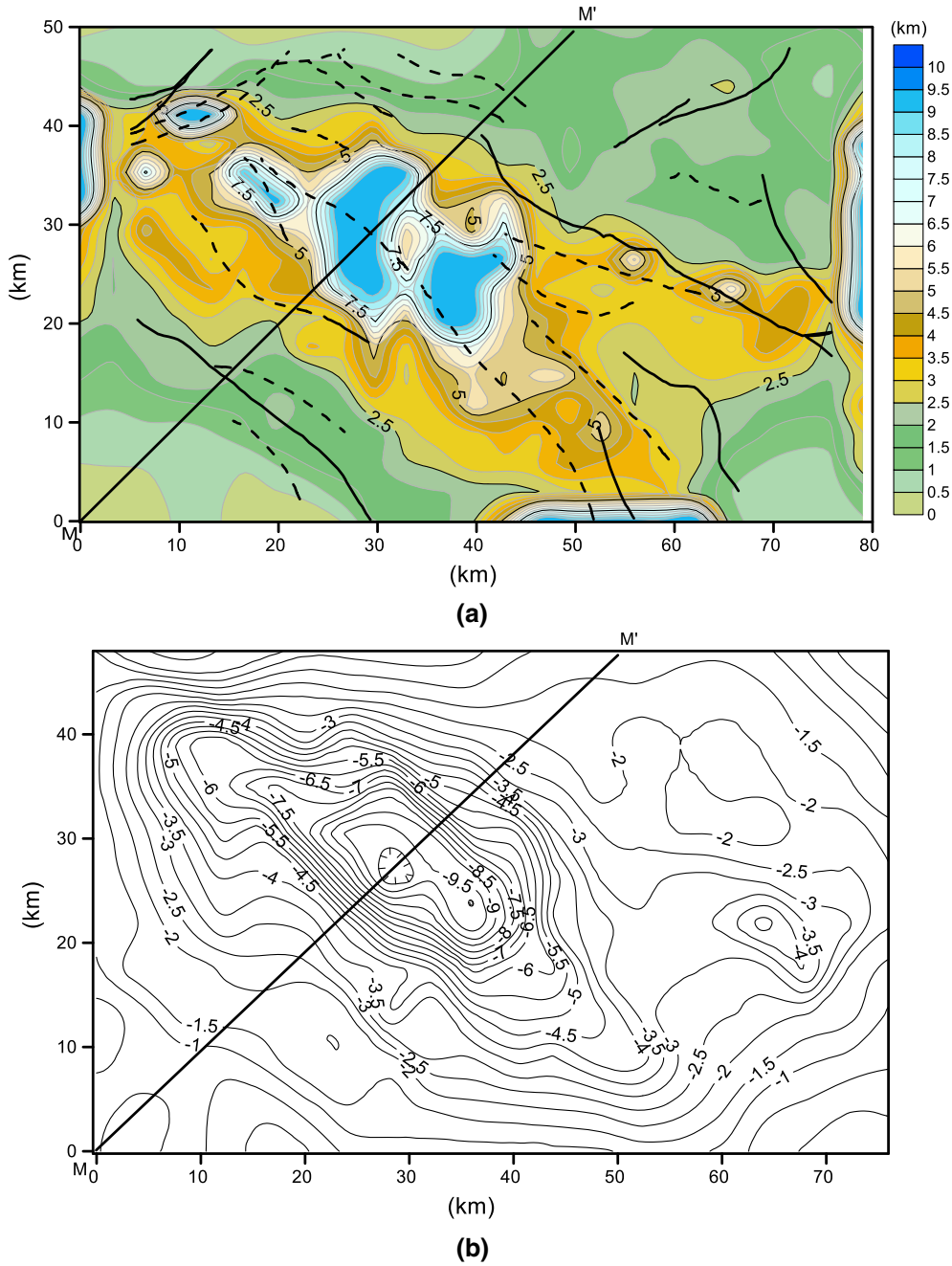


Figure 8

a Estimated structure of the Los Angeles Basin, California, from the present method. The mapped structural features (Yerkes et al. 1965) are also shown, **b** inferred structure of the basin by Chai and Hinze (1988). Note that the depth contours are drawn at 0.5 km contour interval. Cross-sections of the basin along profile, MM', are shown in the lower panel of Fig. 9b

The advantage of the present method is that the anomalies need not necessarily be available at equal intervals. Over and above, the present method is simple, efficient and effective in its implementation when compared to existing 3D interpretation methods that make use of the exponential density model. The

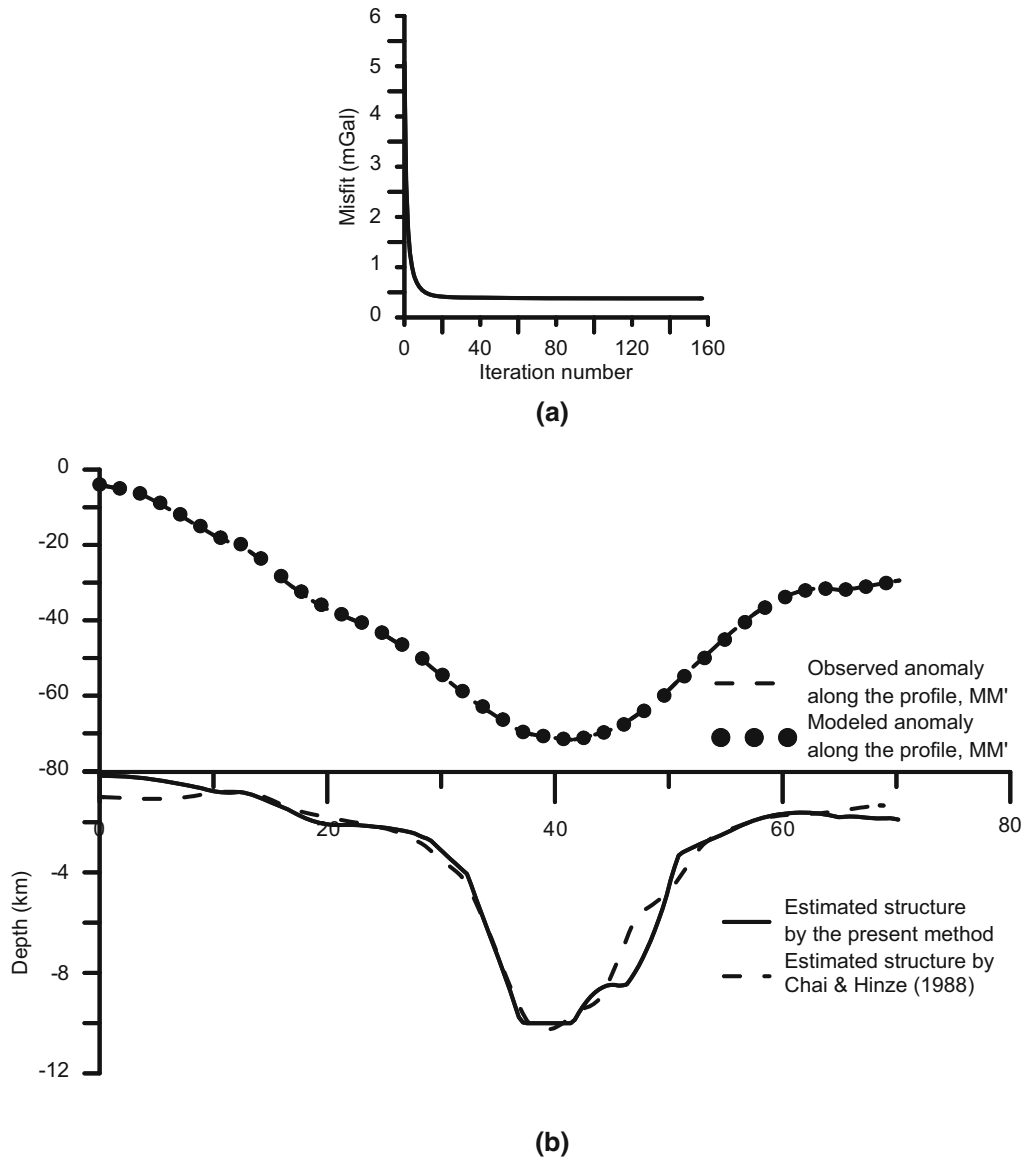


Figure 9
a Variation of misfit against the iteration, **b** observed (dashed) and modeled gravity (solid dots) anomalies (by present method) of the Los Angeles Basin, California, along the profile, MM' (top panel). Cross-sections of the basin along the profile, MM', (lower panel) obtained from the present method and the one by Chai and Hinze (1988)

proposed technique has general applicability to basin delimitation.

Acknowledgements

The authors sincerely thank the reviewers Drs. H Holstein in particular and Coşkun SARI and the Editor Hans-Jürgen Götze for their very useful comments/suggestions and feed back to improve the manuscript as presented.

References

Abdeslem, G. J. (2005). The gravitational attraction of a right rectangular prism with density varying with depth following a cubic polynomial. *Geophysics, 70*(6), J39–J42.

Bal, O. T., & Kara, I. (2012). 3-D Gravity modeling of basins with vertical prisms: Application to Salt Lake region (Turkey). *Journal of the Balkan Geophysical Society, 15*(1), 1–6.

Banerjee, B., & Das Gupta, S. P. (1977). Short note: Gravitational attraction of a rectangular parallelepiped. *Geophysics, 42,* 1053–1055.

Bhaskara Rao, D., Prakash, M. J., & Ramesh Babu, N. (1990). 3D and $2^{1/2}$D modelling of gravity anomalies with variable density contrast. *Geophysical Prospecting, 38,* 411–422.

Blakely, R. J. (1996). *Potential theory in gravity & magnetic applications*. Cambridge: Cambridge University Press.

Cai, H., Xiong, B., & Zhu, Y. (2018). 3D Modeling and Inversion of Gravity Data in Exploration Scale. IntechOpen Limited. https://www.intechopen.com/books/gravity-geoscience-applications-industrial-technology-and-quantum-aspect/3d-modeling-and-inversion-of-gravity-data-in-exploration-scale. Accessed 12 Aug 2018.

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.

Chai, Y., & Hinze, W. J. (1988). Gravity inversion of an interface above which the density contrast varies exponentially with depth. *Geophysics, 53*(6), 837–845.

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. (2009). Automatic gravity inversion for simultaneous estimation of model parameters and regional gravity background: an application to 2D pull-apart basins. *Current Science, 96*(10), 1349–1360.

Chakravarthi, V. (2011). 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., Mallesh, K., & Ramamma, B. (2017). Basement depths estimation from gravity anomalies: Two 2.5D approaches coupled with exponential density contrast model. *Journal of Geophysics and Engineering, 20,* 303–315.

Chakravarthi, V., Pramod Kumar, M., Ramamma, B., & Rajeswara Sastry, S. (2016). 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, 124,* 54–61.

Chakravarthi, V., Raghuram, H. M., & Singh, S. B. (2002). 3D Forward gravity modeling of density interfaces above which the density contrast varies continuously with depth. *Computers & Geosciences, 28,* 53–57.

Chakravarthi, V., Rajeswara Sastry, S., & Ramamma, B. (2013). 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., Shankar, G. B. K., Muralidharan, D., Harinarayana, T., & Sundararajan, N. (2007). An integrated geophysical approach for imaging subbasalt sedimentary basins: Case study of Jam River basin, India. *Geophysics, 72,* B141–B147.

Chakravarthi, V., & Sundararajan, N. (2004a). Automatic 3-D gravity modeling of sedimentary basins with density contrast varying parabolically with depth. *Computers & Geosciences, 30,* 601–607.

Chakravarthi, V., & Sundararajan, N. (2004b). Ridge regression algorithm for gravity inversion of fault structures with variable density. *Geophysics, 69,* 1394–1404.

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 Juan Garcia-Abdeslem (November–December 2005, Geophysics, P. j39–j42). *Geophysics, 71,* X17–X19.

Comacho, A. G., Montesinos, F. G., & Vieira, R. (1997). A three-dimensional gravity inversion applied to Sao Miguel Island Azores. *Journal of Geophysical Research B, Solid Earth and Planets, 102,* 7717–7730.

Cordell, L. (1973). Gravity anomalies using an exponential density-depth function—San Jacinto Graben, California. *Geophysics, 38,* 684–690.

Cordell, L., & Henderson, R. G. (1968). Iterative three-dimensional solution of gravity anomaly data using a digital computer. *Geophysics, 33*(4), 596–601.

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*(1–2), 141–153.

Crosby, A. G., McKenzie, D., & Sclater, J. G. (2006). The relationship between depth, age and gravity in the oceans. *Geophysical Journal International, 166,* 553–573.

D'Urso, M. G. (2014a). Analytical computation of gravity effects for polyhedral bodies. *Journal of Geodesy, 88,* 13–29.

D'Urso, M. G. (2014b). Gravity effects of polyhedral bodies with linearly varying density. *Celestial Mechanics & Dynamical Astronomy, 120*(4), 349–372.

Debeglia, N., & Corpell, J. (1997). Automatic 3-D interpretation of potential field data using analytic signal derivatives. *Geophysics, 62,* 87–96.

Feng, J., Zhang, S., & Meng, X. (2016). Constraint 3D density interface inversion from gravity anomalies. *Arabian Journal of Geoscience*. https://doi.org/10.1007/s12517-015-2213-9.

Gallardo-Delgado, L. A., Perez-Flores, M. A., & Gomez-Trevino, E. (2003). A versatile algorithm for joint 3D inversion of gravity and magnetic data. *Geophysics, 68*(3), 949–959.

Gokula, A. P., & Sastry, R. G. (2015). Gravitational attraction of a vertical pyramid model of flat top-and-bottom with depth-wise parabolic density variation. *Journal of Earth System Science, 124*(8), 1735–1744.

Grabowska, T., Bojdys, G., & Dolnicki, J. (1998). Three-dimensional density model of the earth's crust and the upper mantle for the Area of Poland. *Journal of Geodynamics, 25,* 5–24.

Granser, H. (1987). Three-dimensional interpretation of gravity data from sedimentary basins using an exponential density-depth function. *Geophysical Prospecting, 35,* 1030–1041.

Gu, X., Tenzer, R., & Gladkikh, V. (2014). Empirical models of the ocean-sediment and marine sediment-bedrock density contrasts. *Geosciences Journal, 18*(4), 439–447.

Hansen, R. O. (1999). An analytical expression for the gravity field of a polyhedral body with linearly varying density. *Geophysics, 64*(1), 75–77.

Holstein, H. (2002). Gravimagnetic similarity in anomaly formulas for uniform polyhedral. *Geophysics, 67,* 1126–1133.

Holstein, H. (2003). Gravimagnetic anomaly formulas for polyhedra of spatially linear media. *Geophysics, 68*(1), 157–167.

Isik, M., & Senel, H. (2009). 3D gravity modeling of Buyuk Menderes basin in western Anatolia using parabolic density function. *Journal of Asian Earth Sciences, 34*(3), 317–325.

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.

Li, Y., & Oldenburg, D. W. (1998). 3-D inversion of gravity data. *Geophysics, 63*(1), 109–119.

Maxant, J. (1980). Variation of density with rock type, depth, and formation in the Western Canada basin from density logs. *Geophysics, 45*(6), 1061–1076.

McCulloh. J. H. (1960). Gravity variation and the geology of the Los Angeles basin of California. U.S.G.S. Prof. Paper 400B, B320–B325.

Murthy, I. V. R., Rama Rao, P., & Rao, S. J. (1990). The density difference and generalized programs for two-and three-dimensional gravity modeling. *Computers & Geosciences, 16*(3), 277–287.

Nagy, D. (1966). The gravitational attraction of a right rectangular prism. *Geophysics, 30,* 362–371.

Nasuti, A., & Ardestani, E. V. (2007). 3-D Forward gravity modeling of basement interfaces with quadratic density contrast. EGM 2007 International Workshop. http://www.eageseg.org/data/egm2007/Sessione%20B/Poster%20papers/B_PP_07.pdf. Accessed 12 Aug 2018.

Nelson, T. H., & Fairchild, L. (1989). Emplacement and evolution of salt sills in the northern Gulf of Mexico. *Houston Geological Society Bulletin, 32*(1), 6–7.

Okabe, M. (1979). Analytical expressions for gravity anomalies due to homogeneous polyhedral bodies and translation into magnetic anomalies. *Geophysics, 44,* 730–741.

Papp, G., & Kalmar, J. (1995). Investigation of sediment compaction in the Pannonian basin using 3D gravity modelling. *Physics of the Earth and Planetary Interiors, 88,* 89–100.

Paul, M. K. (1974). The gravity effect of a homogeneous polyhedron for three-dimensional interpretation. *Pure and Applied Geophysics, 112,* 553–561.

Pham, L. T., Oksum, E., & Do, T. D. (2018). GCH_gravinv: a MATLAB-based program for inverting gravity anomalies over sedimentary basins. *Computers & Geosciences (in press).* https://www.sciencedirect.com/science/article/pii/S0098300418301730. Accessed 13 Aug 2018.

Pilkington, M., & Crossley, D. J. (1986). Determination of crustal interface topography from potential fields. *Geophysics, 51,* 1277–1284.

Pohanka, V. (1988). Optimum expression for computation of the gravity field of a homogeneous polyhedral body. *Geophysical Prospecting, 36,* 733–751.

Rao, P. R., Swamy, K. V., & Murthy, I. V. R. (1999). Inversion of gravity anomalies of three-dimensional density interfaces. *Computers & Geosciences, 25,* 887–896.

Rezaie, M., Moradzadeh, A., & Kalate, A. N. (2017). 3D gravity data-space inversion with sparseness and bound constraints. *Journal of Mining & Environment, 8*(2), 227–235.

Talwani, M., & Ewing, M. (1960). Rapid computation of gravitational attraction of three-dimensional bodies of arbitrary shape. *Geophysics, 25*(1), 203–225.

Tenzer, R., & Gladkikh, V. (2014). Assessment of density variations of marine sediments with ocean and sediment depths, *The Scientific World Journal*, https://www.hindawi.com/journals/tswj/2014/823296/. Accessed 08 Aug 2018.

Tsoulis, D. (2000). A note on the gravitational field of the right rectangular prism. *Bollettino di Geodesia e Scienze Affini, 1,* 21–35.

Wu, L., & Chen, L. (2016). Fourier forward modeling of vector and tensor gravity fields due to prismatic bodies with variable density contrast. *Geophysics, 81*(1), G13–G26.

Yerkes, R.F., McCulloh, T. H., Schoellhamer, J. E., & Vedder, J. G. (1965). Geology of the Eastern Los Angeles Basin, California—an Introduction, Geological Survey Professional Paper 420-A, United States Government Printing Office. https://pubs.usgs.gov/pp/0420a/report.pdf. Accessed 10 Aug 2018.

Zhdanov, M. S., & Cai, H. (2013). Inversion of gravity and gravity gradiometry data for density contrast surfaces using Cauchy-type integrals. *SEG Expanded Abstracts*, https://doi.org/10.1190/segam2013-0429.1. Accessed 08 Aug 2018.

(Received August 14, 2018, revised January 11, 2019, accepted January 11, 2019)

rediffmail Mailbox of vcvarthi

Subject: JESS: Your manuscript entitled 3D Spatial Domain Gravity Inversion with Growing Multiple Polygonal Cross-sections and Exponential Mass Density Contrast - [EMID:26e414e424fc5028]

From: Munukutla Radhakrishna <em@editorialmanager.com> on Sun, 10 Jan 2021 08:14:03

To: "Chakravarthi Vishnubhotla" <vcvarthi@rediffmail.com>

You are being carbon copied ("cc:'d") on an e-mail "To" "Ramamma B" ramageophd@gmail.com
CC: "Chakravarthi Vishnubhotla" vcvarthi@rediffmail.com, mradhakrishna.jess@gmail.com, mradhakrishna.iitb@gmail.com

Ref.: Ms. No. JESS-D-20-00031R1
3D Spatial Domain Gravity Inversion with Growing Multiple Polygonal Cross-sections and Exponential Mass Density Contrast
Journal of Earth System Science

Dear Mrs B,

I am pleased to inform you that your manuscript has now been accepted for publication in Journal of Earth System Science.

You will receive the proofs from the journal publishing office after your manuscript is scheduled for publication.

Thank you for submitting your manuscript to this journal.

Yours sincerely,

Munukutla Radhakrishna
Associate Editor
Journal of Earth System Science

Reviewer #1: The authros seem to have addressd my comments to the extent possible, however it still the text appears to be marginally over sized. But the the authors defend that it will pave the way for better clarity tot he readers of J Earth System Science. Otherwise, everything else is Ok and therefore be published as it is.

Reviewer #2: The authors have addressed the comments and issues that I had with the earlier version of the manuscript. Now the manuscript is much improved and it can be accepted for publication in JESS.

****

Our flexible approach during the COVID-19 pandemic

If you need more time at any stage of the peer-review process, please do let us know. While our systems will continue to remind you of the original timelines, we aim to be as flexible as possible during the current pandemic.

This letter contains confidential information, is for your own use, and should not be forwarded to third parties.

Recipients of this email are registered users within the Editorial Manager database for this journal. We will keep your information on file to use in the process of submitting, evaluating and publishing a manuscript. For more information on how we use your personal details please see our privacy policy at https://www.springernature.com/production-privacy-policy. If you no longer wish to receive messages from this journal or you have questions regarding database management, please contact the Publication Office at the link below.

In compliance with data protection regulations, you may request that we remove your personal registration details at any time. (Use the following URL: https://www.editorialmanager.com/jess/login.asp?a=r). Please contact the publication office if you have any questions.