Removing Noise due to Aircraft Motion from Gravity Gradient Data

Wolobah B. Sali

Declaration

I declare that this thesis is my own, unaided work. It is being submitted for the degree of Master of science in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____
(signature candidate)

_____ day of _____ 2008

Abstract

The problem of interpolating an, in general, irregularly sampled gravity gradient field in the presence of noise is considered. The interpolation problem is ill posed. To restore uniqueness we require that the interpolant is consistent with Laplace's equation. The interpolant is tested in a number of numerical experiments and is found to yield results which are far superior to simple polynomial interpolation.

Acknowledgements

I wish to thank my supervisor, Professor Robert de Mello Koch for his constant contribution and support to this research Dissertation. With out his contribution this research would not have been possible.

I would also like to thank the School of Physics, University of Witwatersrand and the family and staff of the African Institute for Mathematical Sciences (AIMS) for their moral and financial support.

And finally thank you to all my family and friends from the School of physics who have seen me through this degree.

Contents

1. Introduction

Sensing gravity gradients may provide a viable approach to geophysical exploration. An airborne gravity gradiometer would allow surveys in remote areas that would not otherwise be accessible. However, noise induced as a result of aircraft motion would, because the gradient signal is so weak, be significant. In this thesis an algorithm is developed that is capable of filtering out this noise due to aircraft motion. In a nutshell, the algorithm exploits the fact that the potential field that is sensed will satisfy Laplace's equation; the noise that corrupts this signal will not. The algorithm has been tested by coding it in MATLAB and using it to process synthetically generated data.

The gravity gradient field is given by taking two derivatives of the gravitational potential. In this thesis we are interested in the zz component of the gradient, obtained by taking two derivatives with respect to z.

The process of generating synthetic data from a given terrain, followed in this thesis, employs an approach in which the terrain is decomposed into a stack of lamina. The analytic response from each layer, treated as an n-sided polygon with a finite thickness, has been derived by Plouff[1]. These analytic responses can then be summed to give the numerical response from the terrain. In section(2) this approach is discussed in complete detail. Of course, this means that the terrain is taken to have a constant density which is not realistic. Density *anomalies* can be introduced by adding extra geological features with a carefully chosen density. In chapter (2.1) the equations used to implement the forward model are derived. These equations allow a computation of the potential, gravitational field response and the gradient response, from an arbitrary n-sided polygon. The equations we obtain are those originally derived by Plouff, as well as a derivative of them, to allow the computation of the gradient. There is an alternative approach, suggested by Talwani and Ewing[3], to the calculation of the gravitational or gradient field sourced by a three dimensional body. This approach provides an alternative numerical integration to that provided by Plouff's method. Talwani and Ewing have very thin layers and deal with the final integration over z numerically. We have verified that they give the same numerical values when implemented on a computer.

In section (2.5) we give a detailed description of the software developed in our implementation of the lamina approach. We calculate both the z component of the gravitational field (g_z) and the zz component of the gravity gradient(g_{zz}). The input to the code is a digital terrain model and a set of observation points. We call these observation points the *observation grid*. The digital terrain model is a collection of heights (measured by the z coordinate) for specific locations (specified by the x, y coordinates). The output from the code are sets of g_z and g_{zz} values for the specified observation points. The code developed implements the polygonal prism lamina approach derived by Plouff and supplements this with analytical expressions for particular anomalies. This allows the user to place ore bodies and other geological features into the otherwise constant density terrain. Each routine is described by explaining what each routine does, what its inputs are and what it outputs. The codes are tested by using Plouff's method to generate numerical results for the response from rectangular prisms. These are found to be in perfect agreement with the known analytic results.

In chapter (3), a model that has been used to describe a natural oil deposit is discussed. Our main interest in this model comes from its obvious relevance for petroleum explorations. The model is called *the Salt Dome Model* in the literature. These geological structures are of special interest as they can create structural cavities and seals which capture and contain hydrocarbons that migrates from oil bearing rocks[2]. These oil bearing rocks accumulate into a dome with a circular shape for each section taken at a different altitude. In section(3.1) the geometric shape used in the implementation of the Salt Dome Model and the gravitational response from this geological structure is discussed. The geometric shape of the anomaly is cylindrical. In section (3.3) we discuss the two gravity gradient components g_{xz} and g_{yz}. We argue that these can be combined to give more information on subsurface density contrasts. We argue this by showing that the derivative with respect to z of the purely vertical component of the gravity gradient tensor (g_{zz}) can be reconstructed for g_{xz} and g_{yz}. In our treatment of the Salt Dome Model, we consider all components of the (second rank) gradient tensor. In section (3) these tensor elements are discussed. In total there are nine components. Only five of these components are totally independent, because the gradient tensor is symmetric $g_{ij} = g_{ji}$ and since we consider the tensor in a source free region, the potential satisfies Laplace's equation implying that the trace of the gravity gradient tensor vanishes[4]. An analytic derivation of the g_{xz} and g_{yz} gradient response from the cylinder is given. Our derivation exploits the cylindrical symmetry of the problem. This is achieved by employing a Bessel function expansion of the Green's function. In section (3.3) the codes developed for the implementation of the Salt Dome Model are discussed. In this section the equation used in developing the salt dome are obtained from the combined gravity gradient calculated and the result obtained is compared to the analytic result obtained from Plouff's and Ewing and Talwani's approximate formula for rectangular prism and lamina respectively.

In chapter (4) the creation of synthetic data is tackled. The goal is to generate data that could have been taken by an aircraft flying in an actual survey. The basic gravitational signal sensed is generated by forward modeling using the density terrain model together with user determined anomalies as input. For the purposes of this report, we have used terrain models that are described by using simple analytic elevation functions (typically Gaussians). Since gravity is a long ranged force, it is also important to determine how much of the surrounding terrain needs to be included in computing the synthetic signal. Intuitively, it is clear that distant terrain will not make a very large contribution. However, this needs to be quantified so that one is able to decide how much of the distant terrain needs to be included in the forward model to obtain an accurate signal. There is a second issue we need to consider: In any real survey, the above ideal signal is corrupted by noise. This noise may be generated internally by the measuring apparatus or may be as a result of aircraft motion. Indeed, local accelerations will be indistinguishable from variations in the gravitational field. The noise generated inside the measuring apparatus will depend on the particular gradiometer used and modeling the precise details of this noise will require a detailed model of instrument operation. For noise due to aircraft motion, there will be both noise from aircraft vibrations (which will depend on the specific aircraft) and noise due to small aircraft accelerations. It is this last component of the noise that we focus on. The main property of the noise that we exploit is the fact that it will not be a harmonic function (it will not satisfy Laplace's equation). We expect that this is true regardless of the gradiometer or aircraft used for the survey. We model the noise as an inaccuracy in the z coordinate of the gradient. Assuming

this inaccuracy δz is small, Taylor expansion of the gradient leads to the model

$$\text{noise associated gradient} = \delta z \frac{\partial g_{zz}}{\partial z} \tag{1.1}$$

where δz is the small changed in the z coordinate and $\frac{\partial g_{zz}}{\partial z}$ is the third derivative of the gravitational potential. See section (4.1) for further details.

In chapter 5 the main goal of this thesis is achieved: a method to remove the noise due to aircraft motion from the sensed gradient response is developed. The problem of interpolating the gravity from the measured values $g_{zz}(\vec{x}_i)$ is an underdetermined problem: we are trying to estimate a function $g_{zz}(\vec{x})$ from a set of discrete values $g_{zz}(\vec{x}_i)$. To restore uniqueness to this problem, we rephrase this estimation problem as the problem of determining that function $\hat{g}_{zz}(\vec{x})$ which (i) is consistent with the measured data values $g_{zz}(\vec{x}_i)$ and (ii) has the smallest norm

$$||g_{zz}||^2 \equiv \int dx \int dy \int dz |\vec{\nabla} \cdot \vec{\nabla} g_{zz}(x,y,z)|^2 \,. \tag{1.2}$$

Functions of zero length with this norm satisfy Laplaces equation. Thus, we are looking for the solution that is consistent with the measured data and that satisfies Laplaces equation. Given the fact that the noise does not satisfy Laplaces equation, we expect that the solution to this estimation problem has the noise removed. In section(5.0.2) computer code which performs this interpolation numerically is developed. Our results show that the method does indeed remove the noise in the sensed gradient signal.

All source code listings are provided as an appendix to this thesis.

The novel content of this thesis, chapter 5, has been submitted for publication to Electronics Letters.

2. Generating Synthetic Data

The problem of generating synthetic data, from a given digital terrain model and a collection of anomalies is solved in this chapter. This data will be used to test the algorithm, developed in this thesis, that removes noise due to aircraft motion from the gradient data. The computation of the expected gradient response is straight forward if we have perfect knowledge of the terrain density. Of course, this is in general not available. For our goal this need not concern us. We simply need to generate a realistic gradient response which can then be used to test our filter.

The ultimate goal of any airborne survey is to reconstruct the detailed density of the terrain from the measured gradient signal. We start this chapter by asking what can be learned about the terrain, given perfect knowledge of the gradient component g_{zz} that we study.

Our approach to the problem of generating synthetic data starts by dividing the terrain into N layers. Each layer is approximated by an n-sided polygon. The analytic response from an n-sided polygon was first obtained by Plouff[1]; we will make use of his result to obtain the response from each later. The total response is then obtained by summing the responses from each layer. Plouff's result is reviewed in this chapter. Following this review we describe our numerical implementation of his formula and describe how our code has been tested.

2.1 General Comments

Newton's description of gravity models the gravitational force as a conservative force: the force is the gradient of the gravitational potential. A *gravimeter* would measure the gravitational field $\vec{g}$, which can be expressed as

$$\vec{g} = -\vec{\nabla}\phi \tag{2.1}$$

where ϕ is the gravitational potential. The negative sign in this last equation is needed to recover the usual statement of the conservation of mechanical energy (phrased in terms of the potential ϕ). In this study we assume that the platform will fly above the earth's surface with a mounted *gradiometer*. The gradiometer will only measure the vertical gradient (g_{zz}) response

$$g_{zz} = -\frac{d^2\phi}{dz^2}\,. \tag{2.2}$$

We will now argue that, given a suitable choice of boundary conditions, we do not need any other components of the gravity gradient tensor to determine the gravitational field and gravitational potential. Consider

$$g_z = -\frac{\partial\phi}{\partial z}\,. \tag{2.3}$$

Integrating both sides of this equation with respect to z, we have

$$\int_{\phi(x,y,z)}^{\phi(x,y,\infty)} d\phi = \phi(x,y,\infty) - \phi(x,y,z) = -\int_z^\infty g_z(x,y,z')dz'\,. \tag{2.4}$$

Although we do not have knowledge of the detailed structure of the source, we certainly do know that the terrain is bounded in the z direction. Thus it is reasonable to require $\phi \rightarrow 0$ as z$\rightarrow \infty$, that is $\phi(x, y, \infty)$=0 and the above equation becomes

$$\phi(x, y, z) = \int_z^{\infty} g_z(x, y, z')dz' \,. \tag{2.5}$$

The arbitrary constant present in any potential has been fixed by requiring that we measure the potential with respect to infinity. Similarly , by using $g_z(x, y, \infty) = 0$ (which follows because the potential goes to zero smoothly at infinity) we have

$$g_z(x, y, z) = \int_z^{\infty} g_{zz}(x, y, z')dz' \,. \tag{2.6}$$

The inversion of the potential fields (that is, the determination of the density distribution from potential field measurement) is not well defined - there are always ambiguities in reconstructing the sources from a given measurement of the potential, taken in a source free region. The ambiguities cannot be resolved using purely mathematical arguments - intuitively we may say some thing like a very massive but distant source give rise to the same field as less massive but nearby source. We will refer to these ambiguities as source ambiguities. The result we have obtained is completely unrelated to source ambiguities. What we are saying is that the potential can be determined uniquely at the flying height of the aircraft given the fact that the acceleration g_z drops off to zero as z increases to infinity, the potential ϕ goes to constant as z increases to infinity and an exact value of the gradient for all values of z greater than or equal to zero is given.

One way to determine the g_z and g_{zz} responses is to convolve the Green's function with the source of the field. We will focus on the case that one has a constant density terrain in mind and simply point out what modifications arise from a non-constant terrain density. The density of the terrain will be denoted by ρ. Let $G(\vec{r} - \vec{r}')$ denote the Green's function used to computed the potential. The equation for g_z is obtained by differentiating the potential with respect to z. The result is

$$g_z(\vec{x}) = \gamma \int d^3x' \frac{\rho(z' - z)}{|\vec{x} - \vec{x}'|^3} \tag{2.7}$$

where γ is the gravitational constant. The integration runs over the digital terrain model. We can also write this equation as

$$g_z(\vec{r}) = -\frac{\partial \phi(\vec{r})}{\partial z} = -\int_{DTM} \frac{\partial G(\vec{r} - \vec{r}')}{\partial z} \rho d^3r' \tag{2.8}$$

where $G(r - r')$ is the Green's function for the gravitational potential. Comparing these last two equations we see that the Green's function which gives $g_z(\vec{r})$ when convolved with the terrain

density is the derivative of the Green's function which gives the potential when convolved with the terrain density. Of course, this last Green's function is just $\frac{1}{|\vec{r}-\vec{r}'|}$. For the case of the gravity gradiometer, which measures the vertical component of the gradient, the above equation can again be differentiated to give the gradient response as

$$g_{zz} = \frac{\partial g_z}{\partial z} = \int_{DTM} \frac{\partial^2 G(r-r')}{\partial z^2} \rho d^3 r', \tag{2.9}$$

where $\vec{r}$ is the observation point and $\vec{r}'$ is the point where the source is located. This last integral is what we evaluate numerically. We will divide the terrain into N parts so that it becomes a stack of lamina. In this way, we replace the integral over the z coordinate with a summation. However, in this approach we need to integrate over the x, y and z coordinates for each layer in the stack. This volume integral over a layer of the terrain is given by Plouff's formula, which is derived by approximating each layer by an n-polygon.

2.2 The Derivation of Plouff's Formula

In this section we will start by reviewing the approach of Talwani and Ewing[3], which preceded Plouff's method[1]. Talwani and Ewing represent the three dimensional terrain that we integrate over by a sequence of thin contours[3]. Each contour is then approximated by a horizontal irregular n-sided polygonal of thickness dz. Each contour line can be approximated to an arbitrarily good accuracy by making the number of sides n sufficiently large. The contribution to the total gravitational response of the terrain, due to each thin layer can be determined analytically at any point external to the terrain. By interpolation a continuous curve can be obtained which relates the height of the lamina with their gravity response[3]. The total area under this curve gives the total gravitational field sourced by the terrain; it is typically evaluated using a numerical integration.

Consider fig (2.1), which shows a massive body with total mass M. We have shown a contour (ABCDEFGH) located at a depth Z. The origin of our coordinate system is fixed by taking the observation point P to lie at $(0,0,0)$; further, we choose a left handed Cartesian coordinate system with the positive z axis pointing vertically downwards. P' is a point located inside the contour. The method of Ewing and Telwani amounts to replacing this contour with a polygon of thickness dz. In this case, the z component of the gravitational field (g_z) sourced by the polygonal is given by

$$g_z = V dz \tag{2.10}$$

where V is a function giving the gravitational field per unit thickness of the polygonal layers. To obtain V we need to integrate over the surface of the polygonal. See figure (2.2) for a definition of the quantities that appear in this integration. Telwani and Ewing have shown that this integral can be reduced to two line integrals, both along the boundary of the polygon (ABCDEFGH).

The expression for V obtained by Telwani and Ewing[3] is given as

$$V = \gamma\rho[\oint d\psi - \oint \frac{z}{(r^2+z^2)^{1/2}} d\psi], \tag{2.11}$$

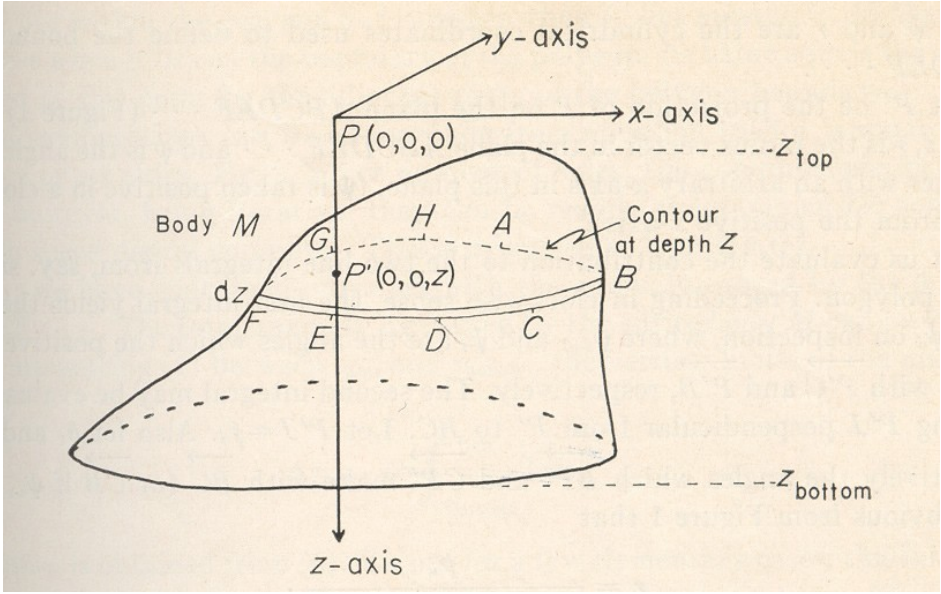

Figure 2.1: A representation of three dimensional contour. This plot is from [3].

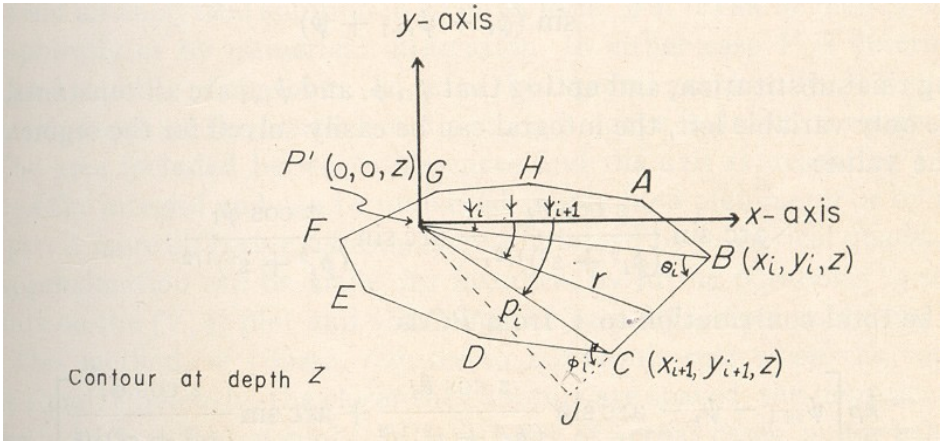

Figure 2.2: Lamina Representation. This plot is from [3].

where γ is the gravitational constant, ρ is the terrain density and z,ψ and r are the cylindrical coordinates used to define the boundary of the polygonal. The integrals over ψ are taken in the clockwise direction. We will consider the contribution from the edge BC is detail. Integrating over this edge corresponds to integrating over ψ from ψ_i to ψ_{i+1}. We have defined ψ as an angle measured with respect to the positive x-axis[3]. Some simple trigonometry now gives

$$r = \frac{p_i}{\sin(\phi_i - \psi_{i+1} + \psi)}. \tag{2.12}$$

In this equation we have assumed that $\psi_{i+1} < \psi_i$. Inserting this into the second term in the expression for V and integrating with respect to ψ we obtain (p_i, ϕ_i and ψ_{i+1} are constants)

$$\arcsin \frac{z \cos\theta_i}{(p_i^2 + z^2)^{1/2}} - \arcsin \frac{z \cos\phi_i}{(p_i^2 + z^2)^{1/2}}. \tag{2.13}$$

The total contribution of the edge BC to V is now easily given as

$$k\rho[\psi_{i+1} - \psi_i - \arcsin \frac{z \cos\theta_i}{(p_i^2 + z^2)^{1/2}} - \arcsin \frac{z \cos\phi_i}{(p_i^2 + z^2)^{1/2}}]. \tag{2.14}$$

The total V sourced by the n-sided polygon is now given by a summation over all of the sides of the polygon. Our final formula for V is

$$V = k\rho \sum_{i=n}^{n} [\psi_{i+1} - \psi_i - \arcsin \frac{z \cos\theta_i}{(p_i^2 + z^2)^{1/2}} - \arcsin \frac{z \cos\phi_i}{(p_i^2 + z^2)^{1/2}}]. \tag{2.15}$$

We can eliminate all dependence on the angles ψ_i. Indeed,

$$\sum_{i=1}^{n} (\psi_{i+1} - \psi_i) = 2\pi \tag{2.16}$$

when P' lies within the polygon. This sum vanishes when P' lies outside the polygon. As explained above, to obtain the total gravitational field sourced by the terrain, we simply integrate with respect to z

$$g_z = \int_{z_{bottom}}^{z_{top}} V \, dz \, . \tag{2.17}$$

Except for a few special cases, the above integral can not be performed analytically and hence we do not obtain a solution in closed form. The above integral is however easily handled using numerical methods. As with any numerical method, there will be some numerical error in the final solution. This method has the disadvantage that this error becomes large for fields computed within or near one of the lamina. The inaccuracy that results from the use of Telwani and Ewing's method can be overcome, as Plouff has shown[1], by fattening each layer to some finite thickness and then integrating over the resulting layer in the z direction. The method works because it is in fact possible to obtain an exact expression for the gravitational field from the resulting three dimensional polygonal prism! Plouff's method simply replaces Ewing and Telwani's thin polygons by fattened layers.

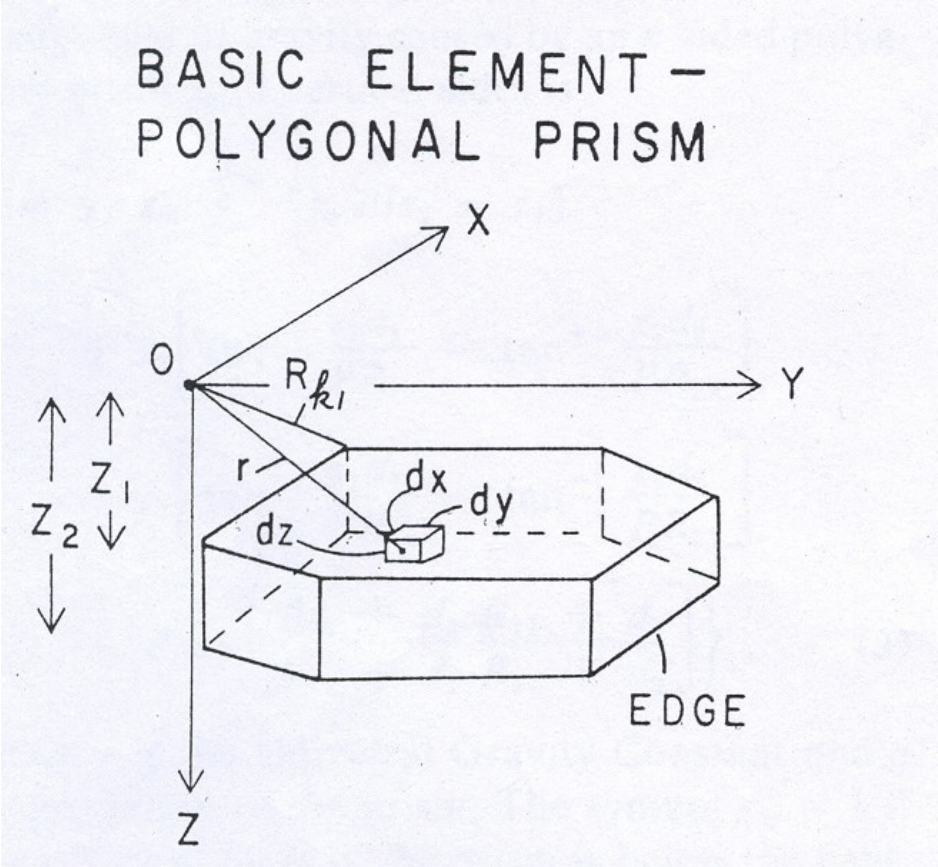

Figure 2.3: One of Plouff's Polygon Prisms. This figure is from [1].

As we have just described, the layers in Plouffs method are polygonal prisms. An example of such a prism is given in figure (2.3). The gravity response from the polygonal prism can be expressed as a sum of contributions from each edge of the n-sided prism, which is the same structure as we saw in the derivation of Ewing and Telwani reviewed above. In figure (2.4) we show a single edge together with the definition of various geometrical parameters needed in the derivation of Plouff's formula. The summation progresses in a clockwise fashion around the individual edges of the prism.

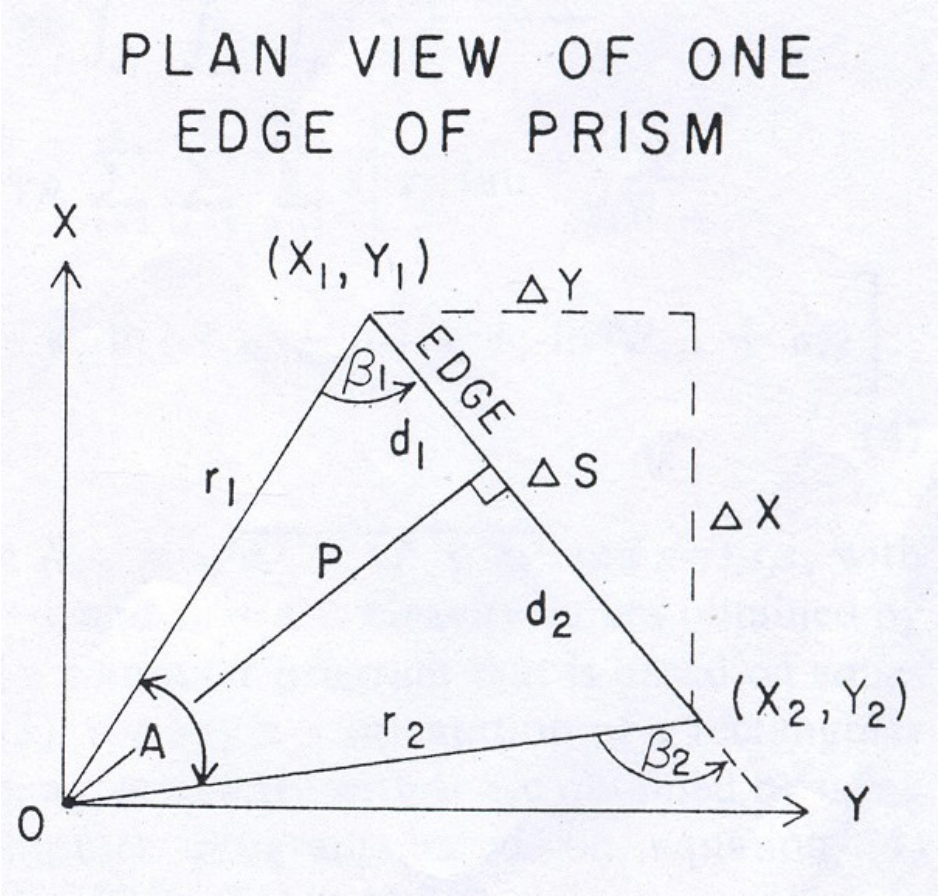

Figure 2.4: One edge of the polygonal prism. This figure is from [1].

In the above diagram subscripts index vertices of the polygon, that is, endpoints of the above edge. The distance $R_{kj} = \sqrt{r_k^2 + z_j^2}$; z_j^2 is the depth to the top of the prism if $j = 1$ and it is the depth to the bottom of the prism if $j = 2$. The radial location of the kth vertex is $r_k = \sqrt{x_k^2 + y_k^2} = \sqrt{d_k^2 + p_k^2}$. A_k is the angle subtended by the k$^{\text{th}}$ edge in the xy-plane. This figure and these definitions are reproduced from [1]. The result of Plouff's analysis replaces the terms appearing in Talwani and Ewing's formula by the term

$$A_i + \arcsin \frac{c_i z}{\sqrt{p_i^2 + z^2}} \tag{2.18}$$

where

$$A_i = \arccos \frac{x_i x_{i+1} + y_i y_{i+1}}{r_i r_{i+1}} = \Delta\psi_i \,, \tag{2.19}$$

$$c_k = -\frac{(x_k \Delta x_k + y_k \Delta y_k)}{r_k \Delta s_k} = \cos\beta_k \,, \qquad s_k = \sin\beta_k \,, \tag{2.20}$$

$$\frac{x_k y_{k+1} - x_{k+1} y_k}{\Delta s_k} = x_k c_k - y_k s_k = p_k \,. \tag{2.21}$$

The parameters Δs_k, Δx_k and Δy_k are all defined in figure (2.4). This last formula can be expressed in an equivalent form as

$$A_i - \arctan \frac{d_i z}{p_i R_{ij}}. \tag{2.22}$$

At this point what we have done amounts to simply rewriting the Telwani-Ewing result. Integrating the above equation with respect to the depth now gives the gravitational response from a polygonal prism. Using the integral identity

$$\int \arctan x \, dx = x \arctan x - \frac{1}{2} \ln[1 - x^2] \tag{2.23}$$

we can perform the required integral to obtain

$$A_i z_j - z_j \arctan \frac{d_i z_j}{p_i R_{ij}} - \ln[\frac{R_{ij} + d_i}{\sqrt{r_i^2 + z_j^2}}]. \tag{2.24}$$

Summing over all edges, we obtain the vertical gravity response due to an n-sided polygonal prism

$$g_z = \gamma \rho s_m \sum_{i=1}^{n} s_p \left[A_i[z_2 - z_1] + z_2 \arctan \frac{z_2 d_i}{p_i} p_i R_{i2} - z_1 \arctan \frac{z_1 d_i}{p_i R_{i1}} \right. \tag{2.25}$$

$$\left. - \ln[\frac{(R_{i1} + d_i)\sqrt{r_i^2 + z_2^2}}{(R_{i2} + d_i)\sqrt{r_i^2 + z_1^2}}] \right]$$

where γ is the universal gravity constant and ρ is the density of the terrain. The symbol $s_m = 1$ if the center of mass of the prism is below the observation point and $s_m = -1$ if the center of mass of the prism is above the observation point. The symbol $s_p = 1$ if p_k is positive and $s_p = -1$ if p_k is negative. The above equation can easily be implemented numerically. An efficient implementation can be obtained by again taking A out of the summation. This sum again gives 2π for observation points located over the interior of the polygon and zero if the observation point does not lie over the interior of the polygon. Further, if the observation point is located over an edge A takes the value π. Finally, if the observation point lies over a vertex of the polygon, A is equal to the interior angle of the intersection of the two edges of the polygon. Plouff's analysis provides an integral formula that accurately gives the gravitational field value anywhere inside or outside the prism. This formula of Plouff's can now be summed over all layers[5] to give the total gravitational field.

To test our numerical implementation of Plouff's method, we have reproduced the known gravitational field sourced by a square prism. This comparison is presented below. In terms of the function $h(x, y, z)$

$$h(x, y, z) = z \arctan(\frac{xy}{zr}) - x \log(y + r) - y \log(x + r) \tag{2.26}$$

where

$$r = \sqrt{x^2 + y^2 + z^2} \tag{2.27}$$

the response from a rectangular prism is given by[5]

$$\begin{aligned} g_z &= -G\rho(h(x-X,y-Y,z-Z) - h(x+X,y-Y,z-Z) - h(x-X,y+Y,z-Z) \\ &\quad -h(x-X,y-Y,z+Z) + h(x+X,y+Y,z-Z) + h(x+X,y-Y,z+Z) \\ &\quad +h(x-X,y+Y,z+Z) - h(x+X,y+Y,z+Z). \end{aligned} \tag{2.28}$$

where X,Y,Z are the lengths of side of the polygonal prism and x,y,and z are input from the contour matrix. The first derivative of this formula yields the gravity gradient $g_{zz}(x)$ sourced by a square prism

$$\begin{aligned} g_{zz} &= -G\rho(H(x-X,y-Y,z-Z) - H(x+X,y-Y,z-Z) - H(x-X,y+Y,z-Z) \\ &\quad -h(x-X,y-Y,z+Z) + h(x+X,y+Y,z-Z) + h(x+X,y-Y,z+Z) \\ &\quad +h(x-X,y+Y,z+Z) - h(x+X,y+Y,z+Z), \end{aligned} \tag{2.29}$$

where

$$H(x,y,z) = -\arctan\frac{yx}{rz} + \frac{xyzr}{x^2y^2+z^2r^2} + \frac{z^3yx}{z^2r^3+y^2x^2r} + \frac{zy}{xr+r^2} + \frac{zx}{yr+r^2}\,. \tag{2.30}$$

2.3 Generation of Gravity and Gravity Gradient Data

A given terrain is typically described by a *digital elevation model*, which quotes a set of heights at a set of locations. One convenient representation of the digital elevation model is in terms of a set of contours of equal height. Indeed, there are by now many standard routines that will generate a set of contours given the digital elevation model. These contours can naturally be turned into a set of polygon vertices for input into Plouff's method. The precise way in which the terrain is discretized will have implications for the numerical errors in the computed gravitational field. Further, we will assume that the terrain has a constant density, which is not realistic and will itself introduce new sources of error. Finally, no digital elevation model is perfectly accurate, so that we will have a third distinct source of error. We will not explore these issues in any detail; our focus is on the noise incurred as a result of aircraft motion. For our purposes it is perfectly reasonable to assume that other sources of noise are negligible.

We have not attempted to apply our results to actual measured digital elevation models. Our terrains are synthetically generated, using nice smooth functions like Gaussians. Is this realistic given that topography can be quite jagged? The signal is sensed at some height above the terrain. Further, the signal that is measured has contributions from the entire terrain - not just jagged cliffs. Thus, even terrain with abrupt changes will give rise to smooth signals. For this reason we feel that considering a synthetic terrain is acceptable for a first study of the problem.

In the remainder of this chapter we give a detailed account of the code we have developed.

2.4 General Comments on Matlab Codes Developed

In developing the code an attempt has been made to separate larger tasks into smaller subroutines. This has been done so that the integrity of each block can easily be verified. It also breaks the code into smaller chunks which can be easily understood and maintained. In documenting the code an attempt has been made to choose variable names that are self explanatory. In addition, the inputs to and outputs from each routine is documented, as well as the dependencies of the routines on each other.

Broadly speaking, the routines developed can be split into two types: those concerned with the forward modeling problem itself and those used to establish the integrity of the developed code. To check the integrity of the code, the numerical responses from a square prism and from a sphere have been compared to the known analytic responses. These match with errors that are of the same order as the numerical precision of the computer on which the code was implemented.

In the next section we will give a detailed discussion of each of the routines that were developed. The source code for all routines are presented in an appendix to this thesis.

2.5 Detailed Discussion of MATLAB Routines

In this section we give a detailed discussion of each MATLAB routine. The goal is to give a clear description of the task each routine performs.

Function $y = cprism(x_0)$

To test the algorithms developed in this thesis, these algorithms were used to generate the gravity response from a rectangular prism. This numerically generated response was then compared to the known analytic response. The function cprism.m computes the analytic response. The input x_0 is a three dimensional vector which contains the coordinates of the point at which the gravity response is to be computed. All coordinate values are measured in meters. The routine outputs the value of g_z in mGals. The Gal is defined as 1 centimeter per second squared.

The routine is a straight forward implementation of the formulas appearing in section 2.2. This routine calls chz.m.

Function $y = cprismg(x_0)$

This routine is also used to test the algorithms developed in this thesis. The function cprism.m computes the analytic gravity gradient response from a square prism. The input x_0 is a three dimensional vector which contains the coordinates of the point at which the gravity gradient response is to be computed. All coordinate values are measured in meters. The routine outputs the value of g_{zz} in Eö. One Eö is 10^{-9} Gals per centimeter.

The routine is a straight forward implementation of the formulas appearing in section 2.2. This routine calls chzz.m.

Function $y = chz(x)$

This function is called by the cprism.m routine and is not intended for use except by this routine. The input variable x is set in cprism.m. This procedure evaluates the function $h(x, y, z)$ of section 2.

Function chzz(x)

This function is called by the cprismg.m routine and is not intended for use except by this routine. The input variable x is set in cprismg.m. This procedure evaluates the function $H(x, y, z)$ of section 2.

Function pot (xdata, ydata, zdata, xobs, dz

The function pot.m evaluates the gravity response g_z, at the observation point $(xobs)$ sourced by a polygon of thickness dz. The boundary of the polygon is specified by the contour whose coordinates are stored in three vectors: x-data, y-data, and z-data. The vectors x-data, y-data and z-data store the x, y and z coordinates of the contour, respectively. In this study we discretize the terrain so that the z-component of the contour is a constant, i.e. so that the lower and upper faces of each polygon are horizontal. Since the contour is closed, the co-ordinates of the first point of the contour coincide with the coordinates of the last point in the contour. There is some error trapping performed in this routine: an "if" statement is used to trap the case in which two contour points coincide. Any segment of zero length does not contribute to the gravity response, but will produce an error if plugged into Plouff's formulae. Apart from this, the code is a straight forward implementation of Plouff's formula.

This routine is called by pgz.m.

Function potg(xdata, ydata, zdata, xobs,dz)

This function evaluates the gravity gradient response g_{zz} at the observation point $(xobs)$ for the polygon of thickness (dz). The input data xdata, ydata, zdata to this function store the boundary coordinate of the polygon, exactly as for pot.m described above. The code is a straight forward implementation of (a derivative with respect to z) of Plouff's original formula.

This routine is called by pgzz.m.

Function pgz(c,xobs,dz)

The function pgz evaluates the gravity response g_z, sourced by a polygon. The field is evaluated at the observation point (xobs). In this routine we assume that the terrain has a constant density.

The contour describing the polygon of thickness dz, is stored in the contour matrix c. The contour matrix is a specific data structure we use in this software. A contour matrix has two rows and N columns; it is responsible for storing the contour lines which together describe the terrain. The different contours are appended end to end as

$$c = \begin{bmatrix} \text{level}_1 & x_1 & \cdots & x_n & \text{level}_2 & x_1 & \cdots \\ \text{pairs}_1 & y_1 & \cdots & y_n & \text{pairs}_2 & y_1 & \cdots \end{bmatrix}.$$

For the matrix shown above, level_1 is the height of the first contour and pairs_1 is the number of points in the first contour. The (x, y) coordinates of these points are shown in the n columns that follow. The second contour's data is then stored and so on until all contours are stored in c.

This routine employs a while statement to process the contour matrix. The limit of this while loop is equal to the number of columns in the contour matrix. The i variable is used as a counter in the while loop. This routine makes a call to pot.m. The input data to pot.m are the coordinates of a single polygon. For a detailed description of how this code extracts the coordinates of the contour points from the contour matrix c, see the description of the pltc.m routine. The hh variable stores the g_z values at the heights given in the variable elevs. Thus the hh variable and the elevs variable have the same size. The response from each polygon is summed to obtain the total response.

Apart from the above description of the code, details and comments are given in the code itself.

Function pgzz(c,xobs,dz)

This function is essentialy the same as pgz.m except that pgzz evaluates the gravity gradient response g_{zz}, sourced by a polygon. See the documentation of pgz.m for further details.

Function pltc(c)

The input data to this code is a contour matrix. This routine produces a plot of the terrain corresponding to the input contour matrix (c). The code starts by storing the number of column in the contour matrix in the variable limit. This variable limit is used in a while loop to process the contour matrix. The integer variable 'i' is used as a counter for the while loop. The while loop runs through the column index of the contour matrix extracting the x,y,z coordinates of each contour. These segments are plotted with the line.m MATLAB routine.

Function genc(xc,yc,zc,lx,ly,lz,N)

This routine generates the contour matrix that is used to model a rectangular prism. The contour matrix generated, is used to verify that the numerical response computed using the lamina method agrees with the analytic response from a rectangular prism. The input data xc, yc and zc specify one of the corners of the prism. The input parameters lx, ly and lz specify the length of the sides of the prism. The parameter N specifies how many elevations the generated contour matrix will use. The bottom of the prism lies at zc and the top at zc+lz. The elevation spacing between

the contours is determined the thickness of the contour dz, where dz=lz/(N-1). See the figure below for a graphical definition of the input parameters.

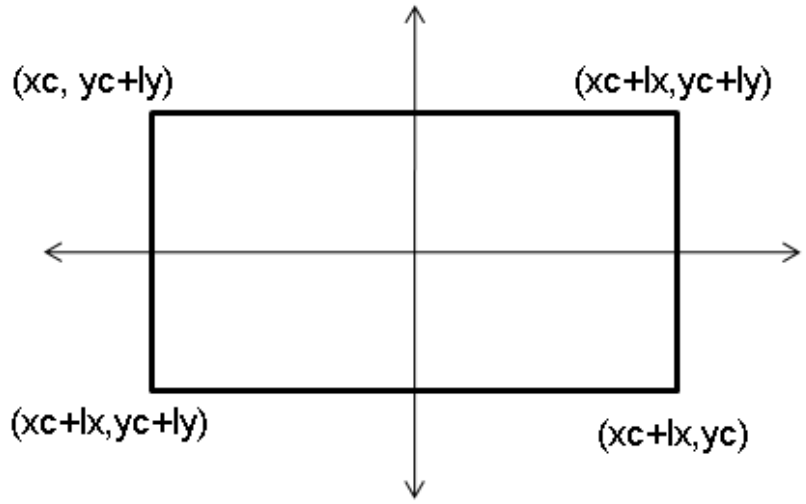

Figure 2.5: Definition of the parameters xc, yc, zc, lx and ly. The above figure shows the prism as seen by an observer looking down the z axis.

Function runc.m

The function of this routine is to check that the routines that generate the numerical gravity response are correct. The routine compares the analytic response from a cubic prism to the numerically generated response, generated using Plouff's formula. In what follows, the "observation grid" is the set of points for which the two responses are computed. The parameter "d" determines the number of points in the observation grid. The variable hd is used to shift the center of the observation grid to the origin.

This code starts by clearing all variables from the workspace and closing all figures that may be open. The contour matrix describing the prism is generated by the routine genc.m. The values of the parameters xc, yc, zc and N are set in runc.m. The pltc.m routine plots the contour matrix generated by genc.m. Next the code calls pgz.m to compute the numerical response. The numerical response is stored in the variable grav1. The analytic response, returned by cprism.m, is stored in the variable grav2. The difference between these responses is stored in the variable grav. The code outputs plots of the contour model of the prism, grav1, grav2 and grav by using the subplot MATLAB command. These plots are reproduced in the next section.

Function runcg.m.

This function is essentially the same as runc.m except that runcg.m checks the numerically generated gravity gradient response. See the documentation of runc.m for further details.

2.6 The Test of the Routines

Each routine defined above has a well defined task. The output of each routine has been verified against simple cases for which the answer can be computed by hand or against analytic formulas. We will not describe these tests in full. Rather we have plotted the outputs from runc.m and runcg.m. From figure 2.6, we see that the error in the numerical response is $\sim 10^{-12}$ (when compared to the leading term), which is the size expected if the noise is due to the finite arithmetic of the computer. The noise in the generated gradient response (see figure 2.7) is of the same size. These results imply that the numerical responses are indeed correct.

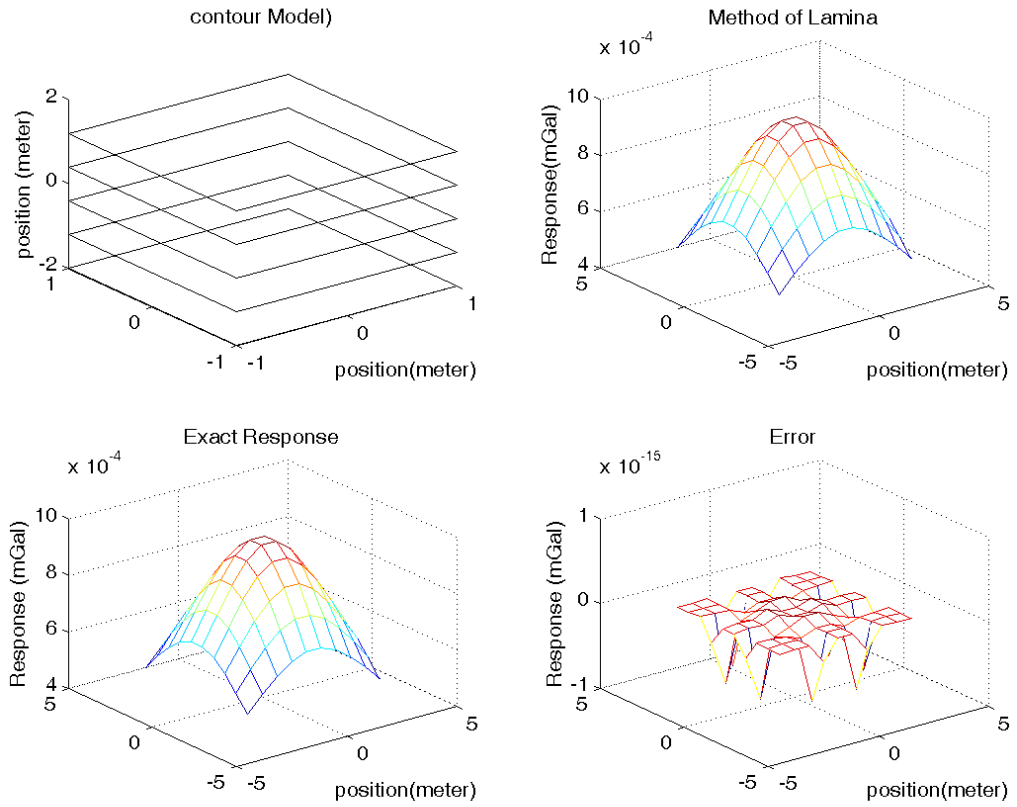

Figure 2.6: Results from the numerical computation of the gravity response from a rectangular prism. The prism is centered at the origin with $l_x = l_y = 2$m and $l_z = 4$m. The observation grid is at a height of 20m. The prism has a density of 0.3 grams per cubic centimeter.

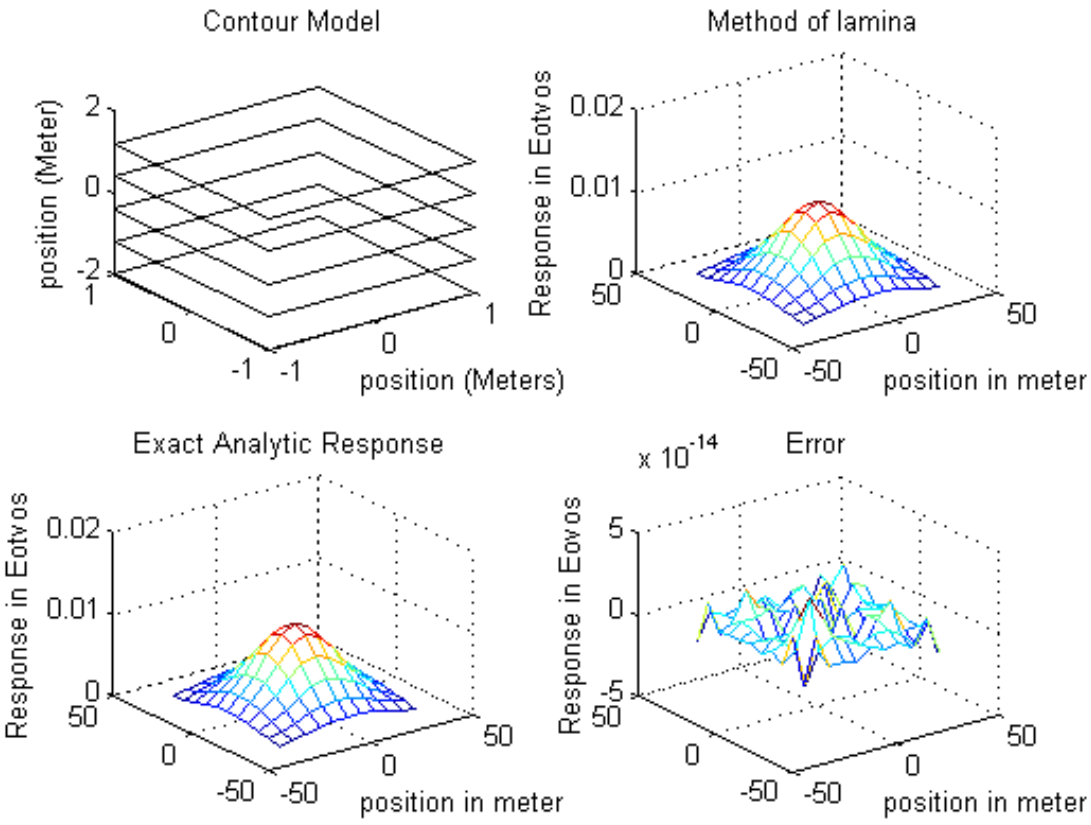

Figure 2.7: Results from the numerical computation of the gravity gradient response from a rectangular prism. The prism is centered at the origin with $l_x = l_y = 2$m and $l_z = 4$m. The observation grid is at a height of 20m. The prism has a density of 0.3 grams per cubic centimeter.

3. The Salt Dome Model

In this chapter we will discuss a geological model known as a salt dome. We discuss a realistic geological model so that we can generate synthetic data that is realistic. This will give the setting in which our filter is tested. The salt dome model is described in the next section. We have stated our interest in developing techniques that can be applied to a gravity gradiometer sensing the g_{zz} component of the gravity gradient. In particular, we are interested in removing noise due to aircraft motion, from the data. This noise signal is in fact determined by both the aircraft motion and the g_{zzz} response from the terrain. Motivated by this observation, we will focus on a study of the g_{zzz} component in this chapter.

3.1 Description of the Salt Dome

Salt domes are of interest for several economic reasons, including

1. The rock salt found in salt domes is mostly impermeable. Salt moving towards the surface tends to bend strata of existing rock, forming pockets between the rock and the salt. Oil accumulation often occurs in these pockets.

2. They are also economically important for the occurrence of free sulfur in some domes of porous limestone cape rocks. Most sulfur is used to generate sulfuric acid that is used in a wide variety of industrial processes, particularly the production of fertilizer. Because of this, sulfuric acid (and hence sulfur) consumption is often regarded as a good index of a nation's industrial development[7].

3. The domes themselves can be excavated for both table salt and the granular material used to prevent roads from freezing over.

Apart from these economic benefits, salt domes are of special interest in the application of gravity sensing techniques to petroleum exploration. This is because they are a near vertical juxtaposition of material with strong density contrasts, providing sizeable anomalies that are particularly susceptible to gravity exploration. Indeed, throughout the history of gravity exploration up to the present, gravity methods have been very effectively applied to determining the presence and depth of domes and delineating their approximate boundaries[5].

Salt domes are relatively simple geological bodies. Their flanks are usually steeply dipping to the vertical or even hanging. Gravity exploration methods are inherently better suited to the detection or delineation of steeply dipping contrasts, of which a vertical fault is the extreme example. As far as their gravitational signal goes, a very accurate model for a salt dome is provided by a vertical cylinder[6]. This is the model adopted for the salt dome in this study. The parameters of the model are the depth to the top of the salt dome (H), the radius (a) of the dome and length (d) of the dome.

3.2 Modeling g_{zzz}

In this section we will discuss some useful properties of the gravity gradient tensor, that will be used in our study of g_{zzz}. In a source free region, the gravitational potential satisfies Laplace's equation

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}=0. \tag{3.1}$$

When written in terms of the gravity gradient tensor, this equation says that the gravity gradient tensor is traceless

$$g_{xx}+g_{yy}+g_{zz}=0. \tag{3.2}$$

This result, together with the fact that the gradient tensor is symmetric, implies that the gravity gradient tensor has only 5 independent components. We can explicitly eliminate g_{zz}

$$g_{zz}=-g_{xx}-g_{yy}. \tag{3.3}$$

Differentiating the above equation with respect to z implies

$$\frac{\partial g_{zz}}{\partial z}=-\frac{\partial g_{xx}}{\partial z}-\frac{\partial g_{yy}}{\partial z}. \tag{3.4}$$

Using the fact that the order in which we take partial derivatives can be changed, we can write this last equation as

$$g_{zzz}=-\frac{\partial g_{xz}}{\partial x}-\frac{\partial g_{yz}}{\partial y}. \tag{3.5}$$

Notice that both sides of this last equation are manifestly invariant under rotations in the $x-y$ plane.

As explained above, we use a cylinder to model the salt dome. Our strategy will be to study g_{xz} and g_{yz}; g_{zzz} is then obtained using the equation we derived above.

3.2.1 The g_{xz} and g_{yz} Gradients Sourced by a Cylinder

In this section we consider a vertical cylinder (i.e. the axis of the cylinder is parallel to the z-axis) of length d, height H (= the depth of the top of the cylinder) and radius a. The gravitational potential is easily expressed as an integral

$$\phi(\vec{x})=-\gamma\int_C \rho(\vec{x}\,')G(\vec{x},\vec{x}\,')d^3x', \tag{3.6}$$

where $\phi(\vec{x})$ is the gravitational potential, $\rho(\vec{x})$ is the density of the cylinder, γ is the gravitational constant and the integration domain C runs over the volume occupied by the cylinder. The Green's function $G(\vec{x},\vec{x}\,')$ is given by

$$G(\vec{x},\vec{x}\,')=\frac{1}{|\vec{x}-\vec{x}\,'|}. \tag{3.7}$$

The cylinder has a rotational invariance about its axis, which is not manifest in terms of the Cartesian x, y, z coordinates we have been using. It makes sense to employ a new set of coordinates which exhibit this symmetry. We leave the z coordinate unchanged. For coordinates transverse to the z-axis we use a radius (r) and an angle (θ) with

$$r = \sqrt{x^2 + y^2}, \qquad \tan\theta = \frac{y}{x}. \tag{3.8}$$

In terms of these new coordinates we have, for example,

$$|\vec{r} - \vec{r}'| = r^2 + r'^2 - 2rr'\cos(\theta - \theta'). \tag{3.9}$$

Our expression for the Green's function in the new coordinates easily follows

$$G(\vec{x}, \vec{x}\,') = \frac{1}{|\vec{x} - \vec{x}\,'|} = \frac{1}{\sqrt{|\vec{r} - \vec{r}\,'|^2 + |z - z'|^2}}. \tag{3.10}$$

We will now derive a Bessel function representation of the Green's function, by expressing it as the Laplace transform of a Bessel function, rewriting this using an addition formula for Bessel functions and then finally Laplace transforming. This will prove to be a very convenient representation in what follows. If in the integral formula

$$\int_0^\infty \exp(-sk) J_o(ak)\, dk \; = \frac{1}{\sqrt{s^2 + a^2}} \tag{3.11}$$

where $J_o(ak)$ is the Bessel function of order zero, we identify $a = |\vec{r} - \vec{r}\,'|$ and $s = |z - z'|$, we obtain

$$\phi(\vec{x}) = -\gamma \int_C \rho(\vec{x}\,') \int_0^\infty \exp(-sk) J_o(ak)\, dk\, d^3x' \,. \tag{3.12}$$

We can trade the integral over k for a sum over a discrete index by using the addition formula for Bessel functions

$$J_o(k|\vec{r} - \vec{r}\,'|) = \sum_{m=-\infty}^{\infty} \exp(im(\theta - \theta')) J_m(kr) J_m(kr') \tag{3.13}$$

where J_m is the Bessel function of order m, and by using the known Laplace transform of a product of Bessel functions

$$\int_0^\infty \exp(-at) J_m(bt) J_m(ct) dt = \frac{1}{\pi\sqrt{bc}} Q_{m-1/2}\left(\frac{a^2 + b^2 + c^2}{2bc}\right), \tag{3.14}$$

where $Q_{m-1/2}(x)$ is the half integer degree Legendre function of the second kind. The potential now becomes

$$\phi(\vec{x}) = -\frac{\gamma}{\pi} \int_C \frac{\rho(\vec{x}\,')}{\sqrt{rr'}} \sum_{m=0}^{\infty} \epsilon_m \cos(m(\theta - \theta')) Q_{m-1/2}(\alpha)\, d^3x', \tag{3.15}$$

where $\epsilon_0 = 1$, $\epsilon_m = 2$ for $m > 0$ and

$$\alpha = \frac{r^2 + r'^2 + (z - z')^2}{2rr'}. \tag{3.16}$$

To obtain the last expression for the potential, we have used the fact that

$$Q_{-m-\frac{1}{2}}(x) = Q_{m-\frac{1}{2}}(x)\,. \tag{3.17}$$

If we now use the double angle formula to expand the $\cos(m(\theta-\theta'))$ factor in the integrand we obtain

$$\begin{aligned}\phi(\vec{x}) &= -\frac{\gamma}{\pi\sqrt{r}}\int_C \frac{\rho(x')}{\sqrt{r'}}Q_{-1/2}(\alpha)\,d^3x' \qquad (3.18)\\ &\quad -2\frac{\gamma}{\pi\sqrt{r}}\sum_{m=1}^{\infty}\cos(m(\theta))\int_C \frac{\rho(x')}{\sqrt{r'}}\cos(m(\theta'))Q_{m-1/2}(\alpha)\,d^3x' \\ &\quad -2\frac{\gamma}{\pi\sqrt{r}}\sum_{m=1}^{\infty}\sin(m(\theta))\int_C \frac{\rho(x')}{\sqrt{r'}}\sin(m(\theta'))Q_{m-1/2}(\alpha)\,d^3x'\,. \qquad (3.19)\end{aligned}$$

We will now specialize to the case of a constant density for the cylinder. The advantage of using the Bessel functions approach is now evident: most of the above terms vanish when integrated over θ'. The gravitational potential reduces to

$$\phi(\vec{x}) = -\frac{\gamma}{\pi\sqrt{R}}\int_C \frac{\rho}{\sqrt{R'}}Q_{-1/2}(\alpha)\,d^3x'\,. \tag{3.20}$$

A useful expression for the half-integer degree Legendre function is

$$Q_{-1/2}(\alpha) = uK(u)\,, \tag{3.21}$$

with $K(\cdot)$ the complete elliptic integral of the first kind and where

$$u = \sqrt{\frac{4rr'}{(r+r')^2+(z-z')^2}}. \tag{3.22}$$

After integrating over θ, the gravitational potential becomes

$$\phi(\vec{x}) = -\frac{2\gamma\rho}{\sqrt{r}}\int_0^a\left[\int_{-H-d}^{-H}\frac{Q_{-1/2}(\alpha)}{\sqrt{r'}}dz'\right]dr'\,. \tag{3.23}$$

In the previous subsection we argued that to obtain g_{zzz} (in which we are ultimately interested) we need to compute g_{xz} and g_{yz}. Thus, we really need to compute

$$g_{xz}(\vec{x}) = \frac{2\gamma\rho}{\sqrt{r}}\int_0^a\frac{\partial}{\partial x}\frac{\partial}{\partial z}\left[\int_{-H-d}^{-H}\frac{Q_{-1/2}(\alpha)}{\sqrt{r'}}dz'\right]dr'\,. \tag{3.24}$$

Now, since the integrand is only a function of $z-z'$, this can further be written as

$$g_{xz}(\vec{x}) = -\frac{2\gamma\rho}{\sqrt{r}}\int_0^a\frac{\partial}{\partial x}\left[\int_{-H-d}^{-H}\frac{\partial}{\partial z'}\frac{Q_{-1/2}(\alpha)}{\sqrt{r'}}dz'\right]dr'\,. \tag{3.25}$$

The integration over z is now trivial

$$g_{xz}(\vec{x}) = \frac{2\gamma\rho}{\sqrt{r}} \int_0^a \frac{\partial}{\partial x} \frac{uK(u)}{\sqrt{r'}} \Big|_{z'=-H-d}^{z'=-H} dr' \,. \tag{3.26}$$

The integrand is only a function of x through its dependence on u. Using

$$\frac{d}{du}(uK(u)) = K(u) + u\left(\frac{E(u)}{u(1-u^2)} - \frac{K(u)}{u}\right), \tag{3.27}$$

where $E(u)$ is a complete elliptic integral of the second kind, we obtain

$$g_{xz}(\vec{x}) = \frac{2\gamma\rho}{\sqrt{r}} \int_0^a \frac{\partial u}{\partial x} \frac{E(u)}{(1-u^2)\sqrt{r'}} \Big|_{z'=-H-d}^{z'=-H} dr' \,. \tag{3.28}$$

This final integral can be performed using the results of [8] to give

$$\begin{aligned} g_{xz} = \gamma\rho \frac{x}{r^2} \Big[& \sqrt{(H+z)^2 + (a+r)^2}\,((2-A)K(A) - 2E(A)) \\ & - \sqrt{(H+d+z)^2 + (a+r)^2}\,((2-B)K(B) - 2E(B)) \Big] \,. \end{aligned} \tag{3.29}$$

where

$$A = \frac{4ar}{(H+z)^2 + (a+r)^2}, \quad B = \frac{4ar}{(H+d+z)^2 + (a+r)^2}, \quad r = \sqrt{x^2+y^2}.$$

A very similar computation gives

$$\begin{aligned} g_{yz} = \gamma\rho \frac{y}{r^2} \Big[& \sqrt{(H+z)^2 + (a+r)^2}\,((2-A)K(A) - 2E(A)) \\ & - \sqrt{(H+d+z)^2 + (a+r)^2}\,((2-B)K(B) - 2E(B)) \Big] \,. \end{aligned} \tag{3.30}$$

3.3 Numerical Analysis

Routines have been written to implement the modeling of two anomalies: the cylinder (chosen for the reasons given in section 3.1) and the sphere (chosen for simplicity). We again build a contour model for these anomalies, and then use the numerical methods and codes developed in chapter 2. In fact, we need a trivial extension of those routines, since we want to model the g_{zzz} response here. This is easily achieved by taking a derivative with respect to z of the formulas obtained in section 2.2.

The code developed could be checked against the known analytic responses from a cylinder (reviewed above) and a sphere. To compare against the numerically generated g_{zzz} response for the cylinder, we have computed the numerical derivative of the analytic g_{xz} and g_{yz} responses obtained above and summed them (see the discussion in section 3.2). The results of this comparison appear in figure 3.1 below.

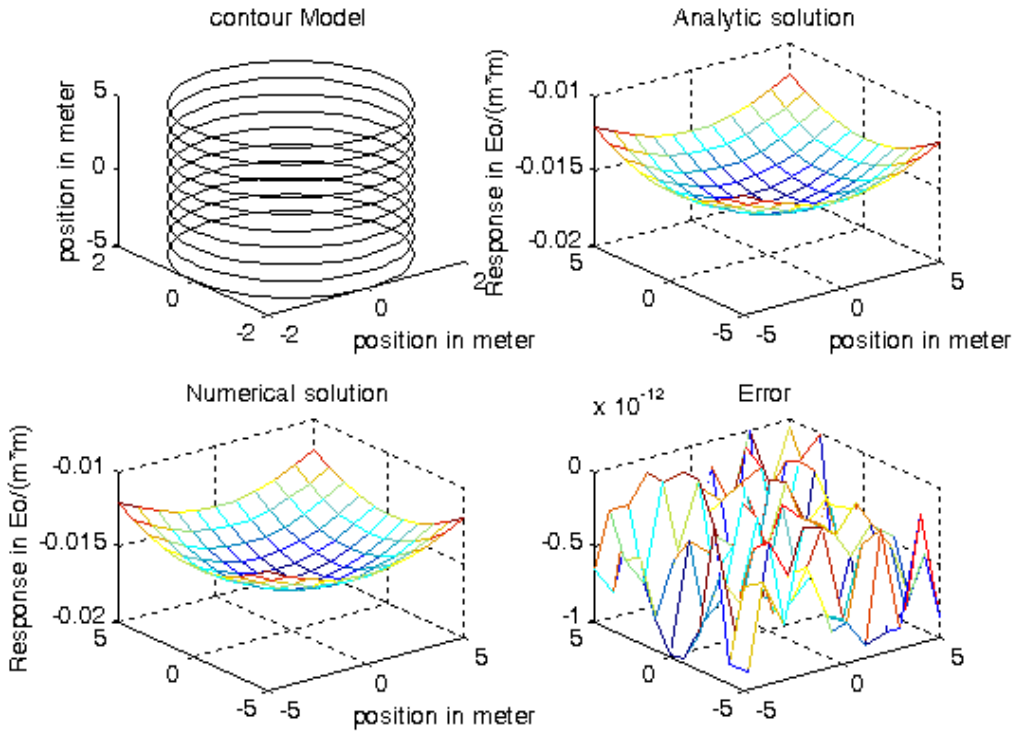

Figure 3.1: The plots of the gravitational response g_{zzz} from a vertical cylinder. For the example above, $a = 5$, $d = 10$ and $H = -5$. The observation grid is at a height of 20m. The cylinder has a density of 0.2 grams per cubic centimeter.

3.4 Matlab Routines for Modeling Anomalies: Cylinder

In this section we list, together with a brief description, the routines used to model the cylinder anomaly.

The function Tx.m and Ty.m

The function Tx.m returns the g_{zx} gradient response from a cylinder; the function Ty.m returns the g_{zy} gradient response from a cylinder. The input parameters to these function are the height (H), thickness (d) and radius (a) of the cylinder as well as the observation grid point (x, y, z) at which the gradient field is sensed. The axis of the cylinder is located at $x = y = 0$. With this choice the cylindrical symmetry of this anomaly corresponds to rotations about the z-axis.

The function der-Tx(x,y,z).m and der-Ty(x,y,z).m

These two functions evaluate the derivative of the output of the Tx.m and Ty.m routines. This derivative is evaluated numerically.

The function Gzzz.m

The function Gzzz.m calls the functions der-Tx(x,y,z).m and der-Ty(x,y,z).m. The output of these routines is summed by Gzzz.m to produce the third derivative of the potential, the g_{zzz} response. The location at which the response is computed is specified by (x, y, z) inside the routine.

The function cgenv.m

This function generates the contour matrix describing a cylinder. The input parameter R specifies the radius of the cylinder. The parameter N specifies how many elevations the generated contour matrix will have. The axis of the cylinder is placed parallel to the z-axis at the origin of the xy plane. This routine plots the contour model using the Pltc.m routine.

The function pggzzz.m

The pggzzz.m routine is responsible for computing the gravity gradient response from an arbitrary terrain model. The input to the routine is a contour matrix c describing the terrain. This function makes a call to the subroutine potgg.m which evaluates the gravity response from a specific polygon by using (a derivative of) Plouff's formula. The input data to this routine includes the thickness dz of each layer as well as the observation grid points.

The function runcgggz.m

This code starts by clearing all variables and closing all open figures, in the work space. The routine starts with a call to the cgenv.m routine to generate the contour matrix corresponding to the cylinder. This contour model is plotted using the pltc.m function. An observation grid is then defined. The routine calls the Gzzz.m function which computes the g_{zzz} response from the cylinder, at the points belonging to the observation grid. The pggzzz.m function is then called on to compute the same response using Plouff's formula. The numerically generated and anayltic responses are plotted and compared.

For more details consult the matlab source code listing in the appendix.

3.5 Matlab Routines for Modeling Anomalies: Sphere

The sphere is a very simple anomaly to model - primarily because it has maximal rotational symmetry. Of course, in practice anomalies are never spherical. However, when the depth to the anomaly is larger than its physical dimensions, the sphere is often an accurate model. In this thesis we have compared the numerically generated (again using Plouff's formulas) responses from a sphere to the exact response.

The response from the sphere is most easily obtained by modeling the sphere as a point particle. This leads to the well known gravitational field

$$g_z = \frac{4\pi R^3 \gamma \rho z}{3r^3} \tag{3.31}$$

where $r = \sqrt{x^2 + y^2 + z^2}$ is the distance from the observation point to the center of the sphere, and $\frac{4}{3}\pi R^3 \rho$ is the mass of the sphere. The gradient responses can be obtained simply by differentiating this result. For example, we obtain in this way the g_{zzz} response

$$g_{zzz} = 4\pi\rho\gamma R^3 \left(\frac{5z^3}{3r^7} - \frac{3z}{r^5} \right) \tag{3.32}$$

Our code compares this result to the numerically generated response.

The matlab functions used in this study are described in the next section.

3.5.1 The Functions for the calculation of the Gradient Response from a Sphere

The function cvgg.m

This function implements the analytic g_{zzz} response from a sphere. The input data to the function are the radius R of the sphere, the density ρ of the sphere and the coordinates of the observation point (x, y, z). The output from this function is the gradient response g_{zzz}.

The function genvgg.m

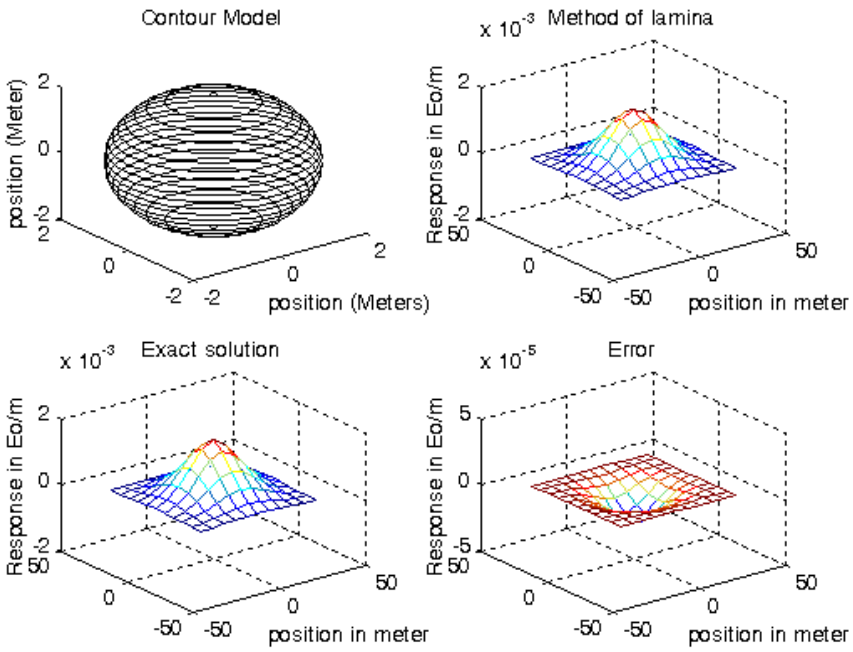

Figure 3.2: The plots of the gravity gradient response from a sphere. The sphere is centered at the origin and has a radius of 2m. The observation grid is at a height of 20m. The sphere has a density of 0.3 grams per cubic centimeter. The error is at the level of a percent of the true signal. This is much higher than the errors for the cylinder or the rectangular prism and is a consequence of the fact that, for the sphere, the contour model is not a perfect representation.

This function generates the contour matrix for the sphere. The sphere is centered at the origin and has radius specified by the parameter R. The parameter N specifies how many elevations the generated contour matrix will have. This function calls on the pltc.m routine to plot the contour model for the sphere.

The function pgzzz.m

This function computes the third derivative of the gravitational potential sourced by the sphere described in the contour matrix generated by genvgg.m, by calling the function potgg.m. The input data to this function are the contour matrix data c, the observation grid point $xobs$ and the thickness of the layers which discretize the sphere dz.

The function potgg.m

This function evaluates the third derivative with respect to z of the gravitational potential (g_{zzz}). The gradient response is computed at the observation grid point (xobs) for the polygon of thickness dz. The polygon vertices are stored in the xdata and ydata vectors; the elevation of the polygon is stored in zdata.

The function runvgg.m

This code starts by clearing all variables and closing all open figures, in the work space. The routine then makes a call to the genvgg.m routine to generate the contour matrix corresponding to the sphere. This contour model is plotted using pltc.m function. An observation grid is then defined. The routine calls the cvgg.m function which computes the g_{zzz} response from the sphere, at the points belonging to the observation grid. The pgzzz.m function is then called on to compute the same response using Plouff's formula. The numerically generated and analytic responses are plotted and compared.

4. The Creation of Synthetic Data

All mass gravitates so that the gravity (gradient) signal of any particular anomaly will be a very weak signal hidden in a huge background signal. Thus, in any airborne gravity survey, it is very important to characterize all the possible sources of noise in the sensed signal. One very important source of noise will be the noise induced by the aircraft motion: upward and downward accelerations of the aircraft will be indistinguishable from the gravitational field that is being sensed. When we discuss noise in this chapter, it is this aircraft motion induced noise, for example, that we have in mind.

The goal of this section, is to produce a model for the data recorded by an airborne gradiometer, which senses the g_{zz} gravitational gradient. There are two ingredients to the signal: the field set up by the terrain over which the instrument flies and noise sensed along with the data. We will discuss these two ingredients separately.

When modeling the terrain, one needs a model for the landscape which comprises of a set of heights as a function of the location of the aircraft $z(x, y)$ and a model for the density of the terrain. We will typically focus on rather flat terrains with a few "hills". For these terrains $z(x, y)$ can be modeled as a sum of Gaussians, which simplifies the analysis. In addition, for the purposes of this thesis, it is sufficient to take a constant density $\rho(z, y, x) = \rho_c$. Even after assuming a simple expression for the terrain $z(x, y)$ and a definite density $\rho(z, y, x)$, an important issue remains: gravity is a long ranged force. When computing the gradient response, we need to convolve the relevant Green's function with the terrain. This is numerically expensive. How much of the distant terrain must be included in this convolution? Recall that the gravity response from an infinite sheet with constant density is a constant, so that the gradient response from the same sheet is zero. Thus, we know that we need only convolve out to some largest radius. The precise value of this radius would be set, in practice, by the accuracy required to sense the anomalies the survey is looking for. In this thesis, this particular point is of no real consequence and we have simply convolved out to roughly 5 thousand kilometers. In the figure 4.1 below, we have shown the gradient response from a flat terrain. Notice that the signal is zero to machine precision. This is exactly what one expects for a flat terrain.

The detailed process used to generate these plots is easily summarized: Given the analytic model $z(x, y)$ for the terrain, a contour model is generated and the contours are inputed to Plough's algorithm together with a constant density $\rho(z, y, x) = \rho_c$.

In the plot 4.2 below, the gradient response from the above flat terrain and a single Gaussian hill are included. It is clear that the signal from the hill completely dominates the signal from the flat terrain. Thus, in terms of the size of the signal (the scale set by the Gaussian hill) the flat terrain is indeed sourcing a signal which is zero. This is a clear indication that enough of the distant terrain has been convolved. We will not delve any deeper into this particular issue.

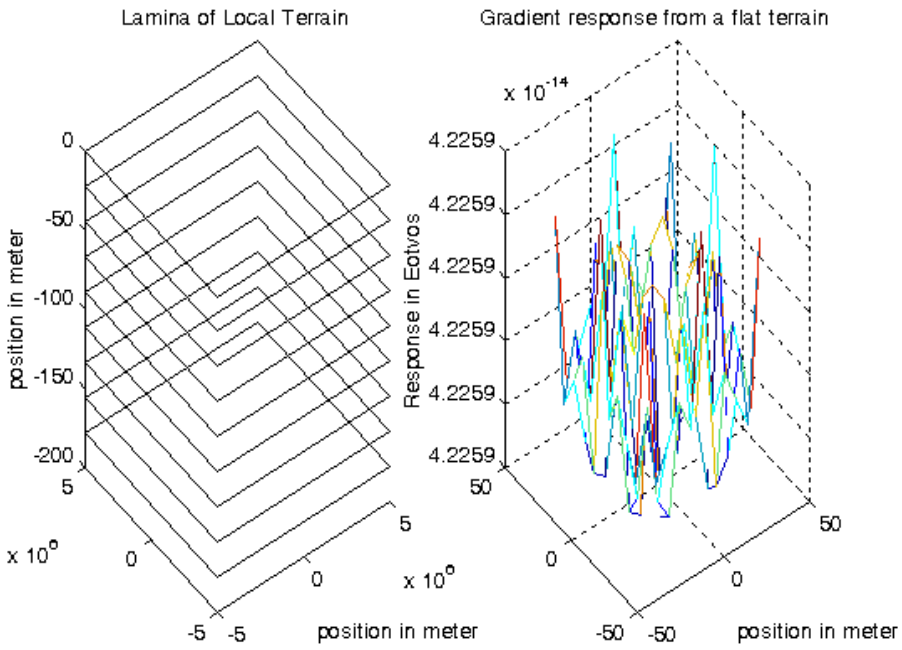

Figure 4.1: The gravity gradient response from a flat terrain.

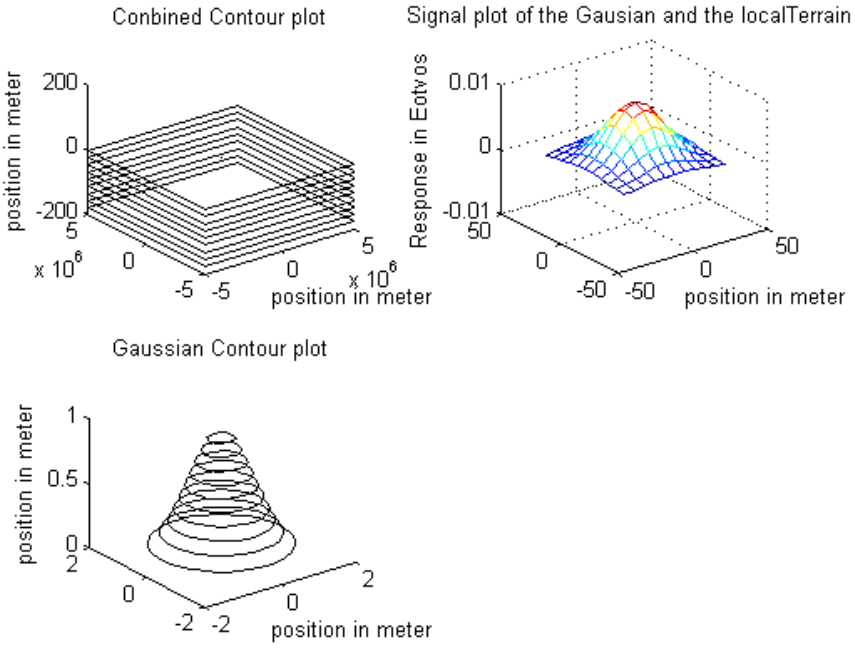

Figure 4.2: The gravity gradient response from a Gaussian hill in a flat terrain.

4.1 The Noise Signal

In this section we are going to develop a model for the noise due to aircraft motion in the sensed gradient signal. Our model is extremely simple and we have not been able to test it against any real data. The main feature of the noise that our model captures is that it is not a harmonic signal, that is, the noise does not satisfy Laplace's equation. Although our model might not represent a very realistic noise signal, it is clear that the signal reproduced from a more accurate model will not be harmonic either. Our filter, developed in the next chapter, is designed to remove a non-harmonic noise signal from a harmonic data signal. Thus, the model we develop should be perfectly satisfactory for our purposes.

In the discussion below we will use the term "perfect gradiometer" to refer to the instrument that senses the signal with no corruption and the term "real gradiometer" to refer to the instrument that senses the noisy signal.

When interpreting the data sensed from an airborne instrument, one has a set of x, y locations together with a height h of the aircraft. The (x, y) coordinates are measured with a very accurate GPS system. The h coordinate is often measured with respect to the terrain, using optical techniques. This can lead to incorrect elevation measurements, since the optical signal can't see through vegetation. Further, the aircraft's elevation is not perfectly constant so that it may be accelerating while the measurement is taken. We will assume that this noise can be modeled as a fluctuation of the vertical position of the aircraft $h \to h + \delta z$. The fluctuation δz is a random variable. At a particular height (h), the perfect gradiometer records the signal

$$-\frac{d^2\phi}{dz^2}|_h = g_{zz}|_h \tag{4.1}$$

where ϕ is the gravitational potential and the x, y dependence is suppressed. The real gradiometer senses the signal

$$\frac{d^2\phi}{dz^2}|_{h+\delta z} = g_{zz}|_{h+\delta z}. \tag{4.2}$$

The fluctuation δz is assumed to be a small number. In this case, we can Taylor expand the signal sensed by the real gradiometer to obtain

$$\frac{d^2\phi}{dz^2}|_h + \delta z \frac{d^3\phi}{dz^3}|_h + O(\delta z^2). \tag{4.3}$$

Thus, the noise in the signal is given by

$$\delta z \frac{d^3\phi}{dz^3}|_h = \delta z g_{zzz}|_h. \tag{4.4}$$

The function g_{zzz} is given by taking derivatives of the potential ϕ. This is evaluated at a flying height where there are no sources. Consequently, ϕ is a harmonic function in this region. This then implies that g_{zzz} itself is a harmonic function. However, the random variable δz will certainly not be a harmonic function and consequently, the noise (which is a product of the two) will certainly not be a harmonic function.

To improve upon the simple noise model given above, one would need to give a detailed model of the sensing process, which is beyond the scope of this thesis.

5. Removing Noise due to Aircraft Motion

Inverting a set of measured gravity gradients to obtain the terrain density is a severely under determined problem. Indeed, only a finite number of measurements are taken; with these measurements we simply can't expect to uniquely reconstruct a continuous function. Further, even if the gradient is known exactly at the flying height for all locations (x, y), there are still ambiguities: a nearby source can have the same signal as a more massive distant source. It is not possible to uniquely resolve these ambiguities in general. In this chapter, we develop an interpolation technique that allows the signal measured at the discrete locations to be interpolated to a continuous function of (x, y). Of course, this interpolation process itself will also not be unique. Notice that the interpolated function represents the gradient sourced by the terrain, at the flying height, that is, in a source free region. Thus, the interpolated function should satisfy Laplace's equation. We are able to obtain a unique interpolant by requiring that the interpolated signal satisfies Laplace's equation "as well as is possible". We do not consider the problem of using this function to infer detailed information about the terrain. Of course, this step also has to be tackled in any geophysical survey.

The general theory exploited in this section has been developed in [11]. This method has been applied to aeromagnetic surveys in [12].

To formulate the interpolation problem, we will first discuss a "signal generation problem". We imagine that the Laplacian L acts on a height (z) varying signal $g_{zz}(x, y, z)$ to produce a zero mean Gaussian white noise process

$$Lg_{zz}(x, y, z) = u(x, y, z). \tag{5.1}$$

L summarizes our knowledge of the signal creation process: up to noise terms, the interpolated signal satisfies Laplaces equation. The survey yields the values a_{ij} for points which are scattered irregularly in space

$$g_{zz}(x_{ij}, y_{ij}, z_{ij}) = a_{ij}. \tag{5.2}$$

In this last expression, the points $\{x_{ij}, y_{ij}, z_{ij}\}$ comprise the observation grid. The image generation model is not uniquely specified until the boundary values of the signal $g_{zz}(x, y, z)$ are specified. We will impose the boundary condition

$$g_{zz}(x, y, \infty) = 0, \qquad Lg_{zz}(x, y, \infty) = 0. \tag{5.3}$$

It is important to note that no attempt is being made to interpolate in three dimensions. For reasons previously discussed, this is impossible without the assumption of a structural geological model. The parameter z is included because Laplace's equation must be satisfied in three spatial dimensions. The selection of a particular value for z has no effect on the normalized interpolant since it simply fixes the source strength to an arbitrary intensity. No upward or downward continuation of the field occurs, so no corresponding high or low pass filtering effects will be observed as the parameter z is varied. No values at the boundary of the (x, y) plane are specified. As we will see, these are not needed to obtain a unique interpolant.

The operator appearing in the signal generation model (L) is known; the signal $g_{zz}(x,y,z)$ can be specified at certain points and at the boundaries. The process $u(x,y,z)$ - which appears due to the noise sensed along with the signal - is not known, so no unique solution for $g_{zz}(x,y,z)$ exists. If we require that the interpolant should be the solution to the signal generation model corresponding to the "input signal" $u(x,y,z)$ with the minimum norm, we obtain a unique problem. Since our interpolant should satisfy Laplace's equation, we will minimize the following norm

$$\int dx \int dy \int dz |u(x,y,z)|^2 = \int dx \int dy \int dz |Lg_{zz}(x,y,z)|^2 \,. \tag{5.4}$$

The last expression suggests an inner product as follows

$$\langle f, g \rangle_L \equiv \int dx \int dy \int dz (Lf)^\dagger (Lg) \,. \tag{5.5}$$

In the solution of most inverse problems, the minimum norm method is only one of several which produce a unique solution, which are optimal in some sense. However, there is often little or no justification for the choice of a specific optimization method. In the present case our choice is by no means ad-hoc - it selects the solution that is most consistent with Laplace's equation.

The solution to the minimization problem is given by

$$g_{zz}(x,y,z) = \sum_{i,j} b_{ij} K(x_{ij}, y_{ij}, z_{ij}, x, y, z), \tag{5.6}$$

where $K(x',y',z',x,y,z)$ satisfies the following equation

$$L^\dagger L K(x',y',z',x,y,z) = \delta(x-x')\delta(y-y')\delta(z-z') \,. \tag{5.7}$$

Thanks to this last equation, we see that

$$\langle f, K \rangle_L = \int dx' \int dy' \int dz' (Lf(x',y',z'))^\dagger L K(x',y',z',x,y,z)$$

$$= \int dx' \int dy' \int dz' f(x',y',z') \, L^\dagger L K(x',y',z',x,y,z) = f(x,y,z). \tag{5.8}$$

Thus, our Hilbert space with the inner product $\langle \cdot, \cdot \rangle_L$ is a reproducing kernel Hilbert space. K is known as the reproducing kernel. For background on reproducing kernel Hilbert spaces see [13]. In the last argument above, f could have been replaced by the reproducing kernel itself. Thus, the reproducing kernel squares to itself on this space: it is a projection operator which projects any function (belonging to the much larger space L_2) into the reproducing kernel Hilbert space (which is a subspace of L_2). The interpretation of our interpolant is now clear: the reproducing kernel projects the vector of measured potential field values b_{ij} into the subspace of Hilbert space which is spanned by harmonic basis functions.

We now consider the problem of explicitly constructing the reproducing kernel K. Thus, we need to solve the partial differential equation

$$L^\dagger L K(x,y,z) = \delta(x)\delta(y)\delta(z) \,, \tag{5.9}$$

with L the Laplacian

$$L = \vec{\nabla} \cdot \vec{\nabla} \,. \tag{5.10}$$

It is straight forwards to verify that $L = L^\dagger$ so that the partial differential equation for the kernel becomes

$$(\vec{\nabla} \cdot \vec{\nabla})^2 K(x,y,z) = \delta(x)\delta(y)\delta(z). \tag{5.11}$$

This is a partial differential equation with constant coefficients, and hence is most easily solved in Fourier space, where it becomes an ordinary algebraic equation. Our conventions for the Fourier transform are

$$K(x,y,z) = \int dk_x \int dk_y \int dk_z e^{ik_x x + ik_y y + ik_z z} K(k_x, k_y, k_z). \tag{5.12}$$

Using

$$(\vec{\nabla} \cdot \vec{\nabla})^2 K(x,y,z) = \int dk_x \int dk_y \int dk_z e^{ik_x x + ik_y y + ik_z z} \left(k_x^2 + k_y^2 + k_z^2\right)^2 K(k_x, k_y, k_z),$$

and

$$\delta(x)\delta(y)\delta(z) = \int \frac{dk_x}{2\pi} \int \frac{dk_y}{2\pi} \int \frac{dk_z}{2\pi} e^{ik_x x + ik_y y + ik_z z} \,,$$

we easily obtain

$$K(x,y,z) = \int \frac{dk_x}{2\pi} \int \frac{dk_y}{2\pi} \int \frac{dk_z}{2\pi} \frac{e^{ik_x x + ik_y y + ik_z z}}{(k_x^2 + k_y^2 + k_z^2)^2}. \tag{5.13}$$

As it stands, the above kernel is not defined. There is a pole at $k_x = k_y = k_z = 0$; we need to define our prescription for treating this pole before the kernel itself is well defined. Of course, the correct prescription is determined by our boundary conditions. Start by performing the integration over k_z. Factorizing

$$k_x^2 + k_y^2 + k_z^2 = (k_z - i\omega_k)(k_z + i\omega_k), \qquad \omega_k^2 = k_x^2 + k_y^2,$$

we see that there are poles at $\pm i\sqrt{k_x^2 + k_y^2}$. A tedious (but straight forward) calculation gives the residue of the pole at $j\omega_k$ as

$$\frac{e^{-\omega_k z + ik_x x + ik_y y}}{(2\pi)^3} \left[\frac{-i}{4\omega_k^3} - \frac{iz}{4\omega_k^2}\right],$$

and the residue of the pole at $-j\omega_k$ as

$$\frac{e^{-\omega_k z + ik_x x + ik_y y}}{(2\pi)^3} \left[\frac{i}{4\omega_k^3} - \frac{iz}{4\omega_k^2}\right].$$

When $z > 0$ we need to close in the upper half plane giving

$$\frac{e^{-\omega_k z + ik_x x + ik_y y}}{(2\pi)^2} \left[\frac{1}{4\omega_k^3} - \frac{z}{4\omega_k^2}\right];$$

and when $z < 0$ we have to close in the lower half plane giving

$$\frac{e^{-\omega_k z + ik_x x + ik_y y}}{(2\pi)^2}\left[-\frac{1}{4\omega_k^3} + \frac{z}{4\omega_k^2}\right].$$

Thus, after integrating over k_z we find

$$K(x,y,z) = \int \frac{dk_x}{2\pi}\int \frac{dk_y}{2\pi}\,\epsilon(z)\frac{e^{-\omega_k z + ik_x x + ik_y y}}{(2\pi)^2}\left[\frac{1}{4\omega_k^3} - \frac{z}{4\omega_k^2}\right], \tag{5.14}$$

where

$$\epsilon(z) \equiv \frac{z}{|z|}. \tag{5.15}$$

To perform the remaining two integrals, change to polar coordinates

$$K(x,y,z) = \frac{1}{(2\pi)^2}\int_0^{2\pi} d\theta \int_0^{\infty} dk e^{ik\cos(\theta)x + i\sin(\theta)yk - kz}\epsilon(z)\left[\frac{1}{4k^2} - \frac{z}{4k}\right].$$

Now, let

$$a = -ik\cos(\theta)x - i\sin(\theta)y - z\,,$$

and use

$$\int_0^{\infty} dr\frac{e^{-ar}}{r^2} = a\log(a) - a, \qquad \int_0^{\infty} dr\frac{e^{-ar}}{r} = -\log(a),$$

to finally obtain

$$\begin{aligned} K(x,y,z) = \ & \tfrac{1}{4}\tfrac{1}{(2\pi)^2}\int_0^{2\pi} d\theta[\ln[-(-(i\cos\theta x + i\sin\theta y - z))\times[(-(i\cos\theta x + i\sin\theta y - z)) + z)] \\ & -(-(i\cos\theta x + i\sin\theta y - z))]]\,. \end{aligned} \tag{5.16}$$

The final integral must be evaluated numerically to obtain the kernel.

Given the above kernel, all that remains to obtain the interpolant is that we specify the coefficients b_{ij}. The condition

$$g_{zz}(x_{ij}, y_{ij}, z_{ij}) = a_{ij}. \tag{5.17}$$

implies

$$\sum_{pq} b_{pq} K(x_{pq}, y_{pq}, z_{pq}, x_{ij}, y_{ij}, z_{ij}) = a_{ij}. \tag{5.18}$$

This is easily inverted to give

$$b_{pq} = K^{-1}(x_{pq}, y_{pq}, z_{pq}, x_{ij}, y_{ij}, z_{ij})a_{ij}. \tag{5.19}$$

This last equation deserves a few comments. The indices i, j run over the set of points at which the original measurements were made. We assume that this set has a total of N points. The indices p, q run over a different set of N points. Thus, the last two equations above can be read as matrix equations. By $K^{-1}(x_{pq}, y_{pq}, z_{pq}, x_{ij}, y_{ij}, z_{ij})$ we mean the matrix inverse of the $N \times N$ matrix $K(x_{pq}, y_{pq}, z_{pq}, x_{ij}, y_{ij}, z_{ij})$. Now that we have determined the b_{pq}, the interpolant

$$g_{zz}(x,y,z) = \sum_{i,j} b_{ij} K(x_{ij}, y_{ij}, z_{ij}, x, y, z), \tag{5.20}$$

can be evaluated at any points (x, y, z).

5.0.1 Testing the Interpolant

We are making a rather strong claim: our method of interpolation will remove (at least partly) the non-harmonic noise present in the measured gravity gradient field. Is this really true? To test our proposal, we will perform the following numerical experiment:

- A terrain together with a complete geological structure is assumed. The exact gravity gradient computed at the set of locations defining an observation grid. This data is noise free.
- Noise is added to the data. The noise added, as explained in the previous chapter, is given by
$$\delta z \frac{\partial g_{zz}}{\partial z} \equiv \delta g_{zzz}. \tag{5.21}$$
The signal g_{zzz} is computed from the assumed geological structure. The variable δz is a random variable, assigned values with the help of MATLAB's random number generator. The MATLAB routine rand.m is used. It produces uniformly distributed random numbers on the interval $[0, 1]$. We obtain in this way, noisy data.
- The noisy data is used to construct the interpolant described in the last section. The interpolant is evaluated at a set of points defining the signal grid. The interpolant is compared to the exact gravity gradient computed at the set of locations defining the signal grid. The power in the noise corrupting the interpolant is computed and compared to the power of the noise added.

By the "noise power" we mean the root mean square of the noise signal

$$M_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{M})^2}. \tag{5.22}$$

As a warm up, we consider the noise free case ($\delta z = 0$). Further, we assume that the observation grid is sampled with a constant spatial period. Both of these assumptions are unrealistic and will be relaxed. The terrain is taken to be a single Gaussian hill. In figure 5.1 below we have shown the contour model in the top left figure. The top right and bottom left figures show the gravity gradient response. (After adding noise, the top right figure will show the noise free response and the bottom left figure will show the noisy response.) The bottom right figure shows the interpolant.

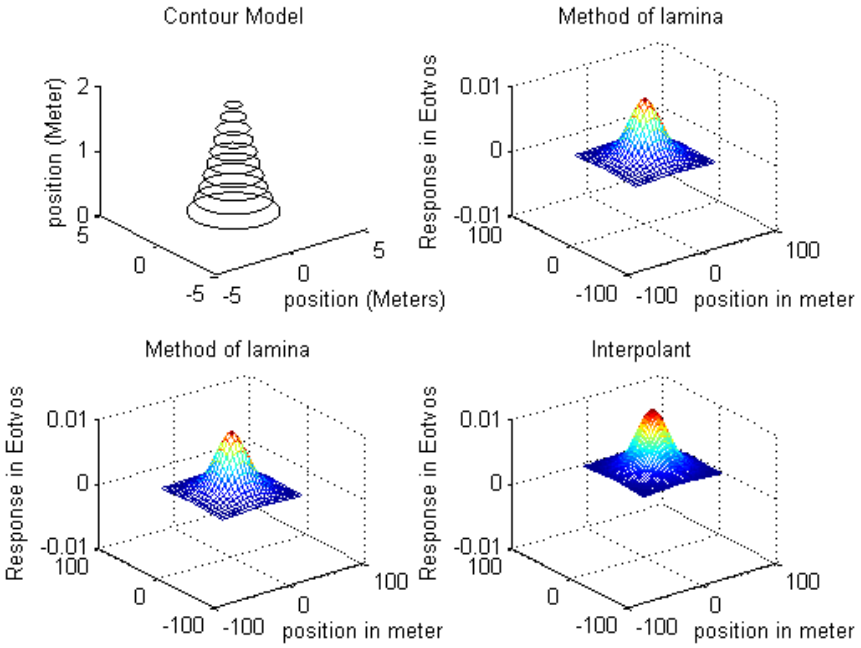

Figure 5.1: We have shown the contour model in the top left figure. The top right and bottom left figures show the gravity gradient response. In subsequent plots, after adding noise, the top right figure will show the noise free response and the bottom left figure will show the noisy response. The bottom right figure shows the interpolant. There is a grid of d^2 points (with $d = 21$) covering the area $-50 \leq (x, y) \leq 50$. The interpolant is used to interpolate the measured values of the signal to an observation grid which is twice as fine as the original grid.

For the smooth response function that we consider here, we would expect a linear interpolant to give an accurate estimate. Explicit construction of the linear interpolant shows that this is indeed the case. In figure 5.2 below, we have compared our interpolant to the linear interpolant. The rms error for both are computed as a function of d, which determines how finely the observation grid samples the $-50 \leq (x, y) \leq 50$ area. A d is increased, both interpolants become more accurate, exactly as expected. The rms error in our interpolant is roughly a factor of 2 smaller than the error in the linear interpolant.

Using a quadratic or cubic interpolant does not improve upon the linear interpolant results. Indeed, the higher order the interpolant used, the more spurious bumps are introduced into the interpolated signal. This is particularly true for noisy data that has been sampled irregularly.

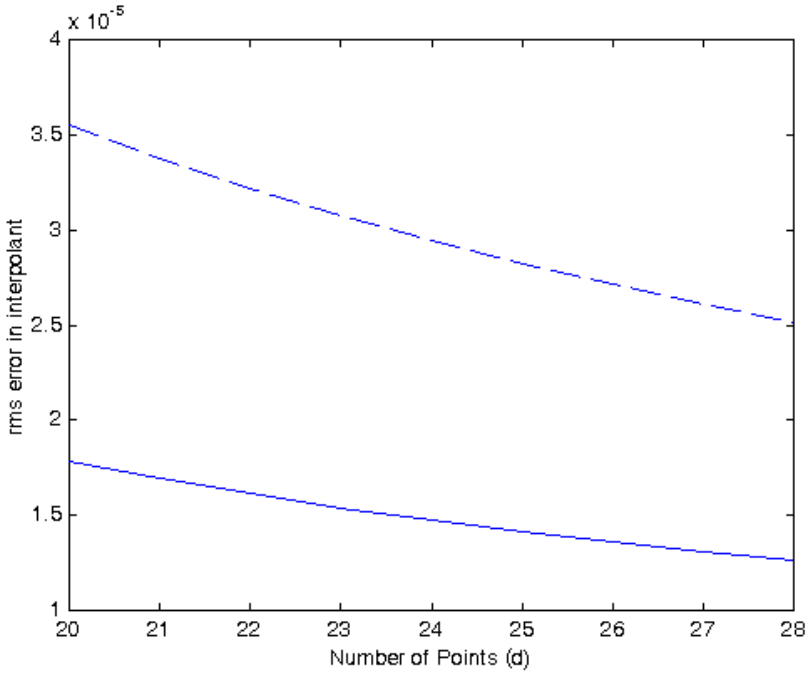

Figure 5.2: The rms error as a function of d for the linear interpolant (dashed line) and our interpolant (solid line).

We will now relax the assumption that that the observation grid is sampled with a constant spatial period. A few comments regarding the linear interpolation are in order. Any three points that do not lie on a line define a plane. Denote the coordinates of these three points by (x_i, y_i, z_i) with $i = 1, 2, 3$. The surface passing through these three points is defined by

$$z = m_1 x + m_2 y + c\,, \tag{5.23}$$

where

$$\begin{bmatrix} m_1 \\ m_2 \\ c \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} .$$

This forms the basis for the linear interpolation that we have implemented.

We will now take a flat terrain together with a set of anomalies that correspond to "ridges" enclosing a "salt dome". The ridges are modeled using rectangular prisms; the salt dome is modeled with a cylinder. See figure 5.3 below. This result shown is rather typical. The linear interpolant identified spurious peaks; the interpolant described in this chapter does not. The rms error in the linear interpolant is typically on the order of a factor of at least 3 greater than the rms error in the interpolant described here. If one decreases the spatial sampling period, to be much finer than the scale of features in the gravitational signal (much finer than what we have in practice), the linear interpolant starts to perform nearly as well as our interpolant.

Finally, introducing noise yields results that are very similar to the results we just obtained when we sampled irregularly. The noisy case will thus not be discussed any further.

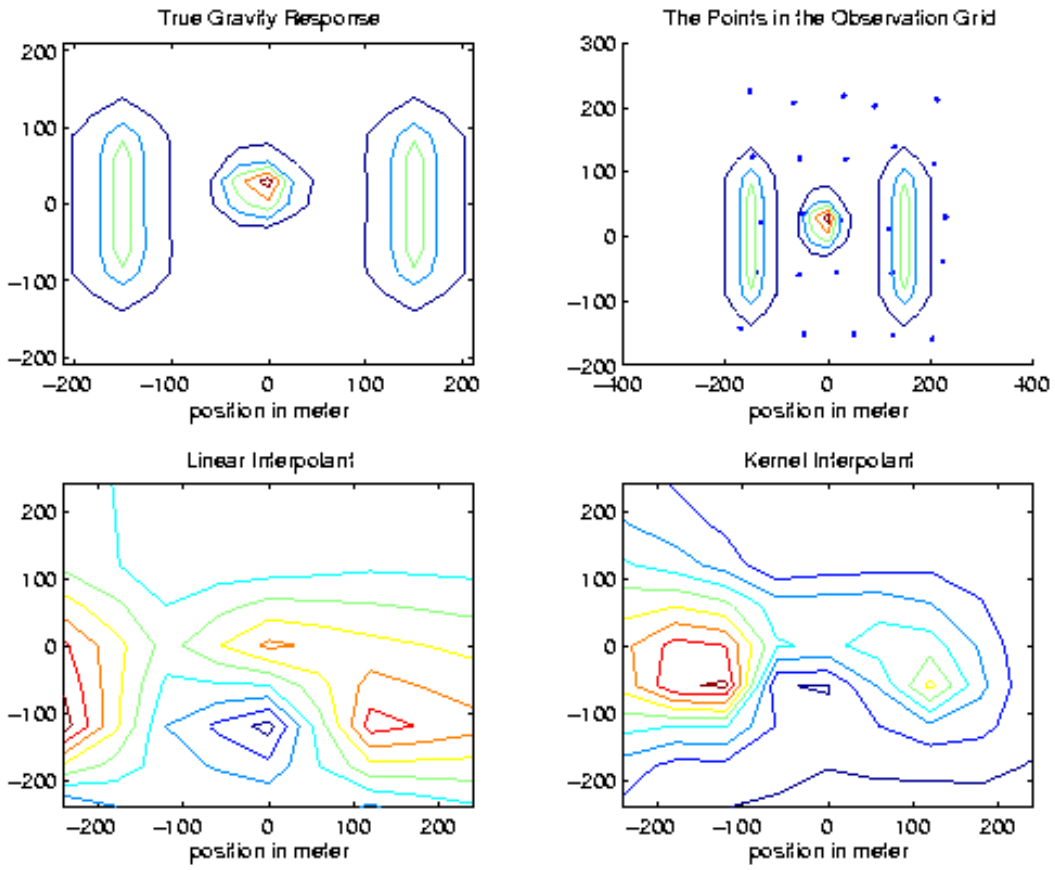

Figure 5.3: The performance of the linear interpolation and our interpolant for irregular spatial sampling. The linear interpolant appears to identify two peaks in the center of the plot; one of these is spurious. The kernel interpolant correctly identifies a single peak.

5.0.2 Matlab Code to generate the Reproducing kernel

There are two MATLAB routines used to test the algorithm developed in this thesis. Since the evaluation of the kernel involves performing an integral over θ, which could not be performed analytically, this integral is performed using Newton's recursive quadrature method. This integral is performed by employing a built in MATLAB function, which makes a call to the routine Gvd.m. Gvd.m is a function of θ, x, y and z. The routine returns the value of the integrand which must be integrated to obtain the kernel - see equation (5.1).

The basic routine which tests the interpolation method is testfilter.m. The routine starts by generating a contour model for the terrain. The contour model is plotted. After this, an observation

grid and an interpolation grid are defined. The observation grid is the set of locations at which we assume a measurement has been taken. The interpolation grid is the set of intermediate points used in the interpolation. Both the observation grid and the interpolation grid contain d^2 points; d is a parameter that is passed to the testfilter.m routine.

The g_{zz} and g_{zzz} responses are computed at the observation grid points. They are stored as a d^2 dimensional vector which is most convenient for the interpolation. Noise is then added to the g_{zz} response. As explained in the previous section, this noise component is a random variable proportional to g_{zzz}. The code then computes the interpolant; this involves the numerical integration of the kernel. Finally, the linear interpolant is constructed.

6. Conclusions

In this thesis we have developed an algorithm that can be used to remove a non-harmonic noise component from a harmonic signal. This algorithm provides a way to interpolate the measured gravity gradient response. In general this problem does not admit a unique solution. We have restored uniqueness by requiring that our interpolant satisfies Laplaces equation.

Testing the interpolant has been an involved process. Indeed, generating synthetic data on which the algorithm could be tested required the development of code to forward model the gravity gradient, given the terrain; this was achieved in chapter 2. Further, the use of this algorithm is to help in the location of mineral deposits by an airborne gravity gradient survey. Thus, it is important to have some understanding of the possible signals coming from mineral deposits. This was achieved in chapter 3, where a salt dome was studied. This particular feature has direct relevance for petroleum exploration. Next a model for the noise needed to be developed. A realistic noise model will involve a detailed model of both the gradiometer and aircraft used and is beyond the scope of this thesis. For our purposes however, the most important feature of the noise is that it is not a harmonic signal. Our noise model entails assuming that the noise can be modeled as an error in the known height of the aircraft. The noise then becomes a random variable (the error in the height) times the g_{zzz} response from the terrain (which is a harmonic signal). The noise signal itself is thus not a harmonic signal and hence the model is good for our purposes.

We have tested our interpolation algorithm by comparing it to linear interpolation in the case that we sample regularly and there is no noise on the data. We have shown that our method out performs linear interpolation in this case. We have also relaxed the assumption of regular sampling and have added noise. In this case too, our algorithm out performs linear interpolation.

7. Appendix: Source code listing of Matlab Codes

The code developed in this report uses MATLAB. In this appendix we list the source code for the routines developed.

The cprism.m routine

```
function yout=cprism(xo);

% yout=cprism(xo);
%
% This  function is used to evaluate the analytic expression
% for the gravity (gz) response from a rectangular prism.
% The result is used as a test for the method of lamina
% algorithm implemented in this study. The center of the prism
% lies at the origin of the coordinate system. xo is the point
% at which the field is observed. The length of the prism is set
% in the routine.

% set the length of the sides of the prism in meters.

X=2; Y=2; Z=4;  X=X/2; Y=Y/2; Z=Z/2;
x=xo(1); y=xo(2); z=xo(3);

% Newton's gravitational constant.

gamma=6.67/1e11;
Grho=300*gamma;

% Evaluate the gravitational response

gz=chz([x-X,y-Y,z-Z]) ...
-chz([x+X,y-Y,z-Z])-chz([x-X,y+Y,z-Z])-chz([x-X,y-Y,z+Z]) ...
+chz([x+X,y+Y,z-Z])+chz([x+X,y-Y,z+Z])+chz([x-X,y+Y,z+Z]) ...
-chz([x+X,y+Y,z+Z]);

yout=Grho*gz*1e5;
```

The cprismg.m routine

```
function yout=cprismg(xo);

% yout=cprismg(xo);
%
% This function evaluates the analytic gravity gradient gzz response
% from a square prism. This result is used to test the method of lamina
% algorithm implemented in this study.  The center of the prism
% lies at the origin of the coordinate system. xo is the point
% at which the field is observed. The length of the prism is set
% in the routine.

% set the length of the sides of the prism in meters.

X=2; Y=2; Z=4; X=X/2; Y=Y/2; Z=Z/2;
x=xo(1); y=xo(2); z=xo(3);

%Newton's gravitational constant.

gamma=6.67/(1e11);
Grho=300*gamma;

% Evaluate the gravity gradient response

gzz=chzz([x-X,y-Y,z-Z]);
gzz=gzz-chzz([x+X,y-Y,z-Z])-chzz([x-X,y+Y,z-Z])-chzz([x-X,y-Y,z+Z]);
gzz=gzz+chzz([x+X,y+Y,z-Z])+chzz([x+X,y-Y,z+Z])+chzz([x-X,y+Y,z+Z]);
gzz=gzz-chzz([x+X,y+Y,z+Z]);

yout=-Grho*gzz*(1e9);
```

The chzz.m routine

```
function yout=chzz(X);

% yout=chzz(X);
%
% This function is called by cprismg.m. The vector X is set in cprismg.
% The chzz function is used to evaluate the gravity gradient response
% gzz from a rectangular prism.

x=X(1); y=X(2);z=X(3);
R=sqrt(x^2+y^2+z^2);
```

```
y1=(x*y*z*R)/(z^2*R^2+y^2*x^2);
y2=(z^3*y*x)/(z^2*R^3+y^2*x^2*R);
y3=(z*y)/(x*R+R^2);
y4=(x*z)/(y*R+R^2);
yout=-atan(y*x/(R*z))+y1+y2+y3+y4;
```

The chz.m routine

```
function yout=chz(X);
% yout=chz(X);
%
% This function is called by cprism.m. The vector X is
% set in cprism.m. The chz function is used to evaluate
% the gravity response gz from a rectangular prism.

x=X(1);,y=X(2); z=X(3);
R=sqrt(x^2+y^2+z^2);
y1=log(y+R);
y2=log(x+R);
yout=z*atan(x*y/(z*R))-x*y1-y*y2;
```

The pgz.m routine

```
function g=pgz(c,xobs,dz);
% g=pgz(c,xobs,dz);
%
% This function evaluates the gravity response (gz) at the
% of the observation point (xobs) sourced by the terrain
% described by a set of polygons, stored in the contour
% matrix c. This routine calls Pot.m to compute the value of
% the gravity response from a single polygon.
% This is then integrated over z to obtain gz.

% the size of the contour
csize=size(c)

% limit is given by the number of columns in the contour matrix
% limit is used in the while loop below
```

```
limit=csize(2)

% initialize variables
hh=[];
elevs=[];
i=1;
pp=0;
zold=0;
while (i < limit)
    % the elevation of the contour.
    z_level=c(1,i);
    % the number of points in the contour
    npoints=c(2,i);
    % next contour start index
    nexti=i+npoints+1;
    % the coodinates of points in contour
    xdata=c(1,i+1:i+npoints);
    ydata=c(2,i+1:i+npoints);
    % compute the gravitational for this polygon
    zdata= z_level*ones(1,npoints);
    if (zold==z_level)
        hh(pp)=hh(pp)+pot(xdata, ydata , zdata, xobs, dz);
    else
        pp=pp+1;
        hh(pp)=pot(xdata, ydata , zdata, xobs, dz);
        elevs(pp)=z_level;
    end %1f
    i=nexti;
    zold=z_level;
end %while

% sum to obtain total response
g=sum(hh);
gamma=6.67/1e11;
Grho=300*gamma;
% normalize the total response
g=-Grho*g*1e5
```

The pgzz.m routine

```
function g=pgzz(c,xobs,dz);
% g=pgzz(c,xobs,dz);
```

```
%
%
% This function evaluates the gravity gradient response (gzz)
% at the observation point (xobs), sourced by the terrain
% described by a set of polygons, stored in the contour
% matrix c. This routine calls Potg.m to compute the value of
% the gravity gradient response from a single polygon.
% This is then integrated over z to obtain gzz.

% the size of the contour matrix
csize=size(c);

% limit is the number of rows in the contour matrix
limit=csize(2);

% initialize variables
hh=[];
elevs=[];
i=1;
pp=0;
zold=0;
while(i < limit)
    z_level=c(1,i);
    npoints=c(2,i);
    nexti=i+npoints+1;
    pp=pp+1
    xdata=c(1,i+1:i+npoints);
    ydata=c(2,i+1:i+npoints);
    zdata=z_level*ones(1, npoints);
    % compute the gradient response from a single polygon
    hh(pp)= potg(xdata,ydata,zdata,xobs,dz);
    i=nexti;
 end %while
% sum to obatin the total response
g=sum(hh);
gamma=6.67/(1e11);
Grho=300*gamma;
% normalize the total response
g=Grho*g*1e9
```

The genc.m routine

```
function cout=genc(xc, yc, zc, lx, ly, lz, N);
% cout=genc(xc, yc, zc, lx, ly, lz, N);
%
% This funtion sets up the contour matrix used to compare
% the numerically generated and analytic responses from a
% rectangular prism.
%
% The input parameters are the corners of the prism, the
% lengths of the sides of the prism and the number (N) of
% elevation contours used in the model. The bottom of the
% polygon prism is at elevation zc and the top is at
% elevation zc+lz.

%  xc = the x coordinate of the corner
%  yc = the y coordinate of the corner
%  zc = the z coordinate of the corner
%  lx = the length of the prism in the x-direction
%  ly = the length of the prism in the y-direction
%  lz = the length of the prism in the z-direction
%  N  = the number of contors to be used

% the x and y coordinates for the closed contour are
x=[xc, xc, xc+lx, xc+lx, xc];
y=[yc, yc+ly, yc+ly, yc, yc];

% the elevation spacing between sucsesive contours.
dz=lz/(N-1);

% the elevation values for the contours
z=zc:dz:(zc+lz-dz);

% intialization of the contour matrix
c=zeros(2,(N-1)*6);

for i =1:(N-1),
    % the number of columns in the contour matrix
    p=(i-1)*6;
    % elevation of z(i).
    c(1,p+1)=z(i);
    c(2,p+1)=5;
    % the number of data points
    in=(p+2):(p+6);
    % set x coordinates
    c(1,in)=x;
    % set y coordinates
```

```
    c(2,in)=y;
end;%for
% The contour matrix is outputed
cout=c;
```

The runc.m routine

```
% runc
%
% This routine checks the numerical forward modelling results.
% The gravity response of a cube is computed using the polygon
% method. This routine then compares the polygon result and the
% known analytic result.

% initialization
clear all ; close all;

% generate the contour matrix corresponding to the cube.
xc=-1;
yc=-1;
zc=-2;
N=6;
c=genc(xc,yc,zc,-2*xc,-2*yc,-2*zc,N);

% generate the elevations
dz=-2*zc/(N-1);

% the contour model is plotted.
subplot(2,2,1)
pltc(c);
title('contour Model)');
xlabel('position(meter)')
ylabel('position (meter)');
zlabel('position (meter)');

%the parameter d sets the observation grid
d=10; hd=(d+1)/2;
for i=1:d,
    for j=1:d,
        xobs=[0.67*(i-hd), 0.67*(j-hd),-6];
        % numerical response
        grav1(i,j)=-pgz(c,xobs,dz);
        % analytic response
        grav2(i,j)=cprism(xobs);
```

```
    end%for j
    x(i)=xobs(1);
end;% for i

% grav is the difference between the analytic
% and numerical responses, i.e. its the error
grav=(grav1-grav2);

subplot(2,2,2);
% plot numerical response
mesh(x,x,grav1)
title('Method of Lamina');
xlabel('position(meter)');
ylabel('position(meter)');
zlabel('Response(mGal)');

subplot(2, 2, 3)
% plot exact response
mesh(x,x,grav2)
title('Exact Response');
xlabel('position(meter)');
ylabel('position (meter)');
zlabel('Response (mGal)');

subplot(2, 2, 4)
% plot error
mesh(x,x,grav)
title('Error');
xlabel('position(meter)');
ylabel('position (meter)');
zlabel('Response (mGal)');
```

The runcg.m routine

```
% runc
%
% This routine checks the numerical forward modelling results.
% The gravity gradient response of a cube is computed using the
% polygon method. This routine then compares the polygon result
% and the known analytic result.
```

```
% initialization
clear all;
close all;

% generate the contour matrix describing the cube
xc=-1;
yc=-1;
zc=-2;
N=6;
c=genc(xc,yc,zc,-2*xc,-2*yc,-2*zc,N);

% the thickness of the polygon.
dz=-2*zc/(N-1);

% the contour model is plotted
subplot(2,2,1);
title('Contour Model');
xlabel('position (Meters)');
ylabel('position (Meter)');
zlabel('position (Meter)');

% d sets the observation grid
d=11; hd=(d+1)/2;
for i=1:d,
    for j=1:d,
        xobs=[6.7*(i-hd),6.7*(j-hd), 20];
        % the numerical gradient response is computed
        grav1(i,j)=pgzz(c,xobs,dz);
        % the analytic gradient response is computed
        grav2(i,j)=cprismg(xobs);
    end; %for j
    x(i)=xobs(1);
end;% for i

% the error
grav=(grav1-grav2);

subplot(2, 2, 2);
% plot the numerical response
mesh(x,x,grav1)
title('Method of lamina');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');
```

```
subplot(2, 2, 3);
% plot the exact response
mesh(x,x,grav2);
title('Exact Analytic Response');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');

subplot(2, 2, 4);
% plot the error
mesh(x,x,grav);
title('Error');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eovos');
```

The Tx.m routine

```
% function h=Tx(x,y,z);
%
% This function returns the gradient xz response from a cylindrical anomaly
% x and y are vectors containing the coordinates of the observation points
% at which the gradient is to be computed. z is the height at which the
% response is required.

% The center of the cylinder lies at (X0,Y0). a is the radius of the cylinder.
% d is the length of the cylinder. H is the height of the top of the cylinder.
% G is Newtons gravitational constant. sigma is the terrain density.
global G sigma H X0 Y0 a d

r= sqrt((x-X0).^2 + (y-Y0).^2);
D=sqrt( (H+z)^2 + (a+r).^2 );
F=sqrt( (H+d+z)^2 + (a+r).^2 );
A=(4*a*r)./(D.^2);
B=(4*a*r)./(F.^2);
[KA,EA]=ellipke(A);
[KB,EB]=ellipke(B);
if r~=0
    h=G*sigma*1e9*( (x-X0)/(r^2) )*D*( (2-A)*KA-2*EA )-F*( (2-B)*KB-2*EB);
 else
  h=0;
```

```
end;
```

The Ty.m routine

```
% function h=Ty(x,y,z);
%
% This function returns the gradient yz response from a cylindrical anomaly
% x and y are vectors containing the coordinates of the observation points
% at which the gradient is to be computed. z is the height at which the
% response is required.

% The center of the cylinder lies at (X0,Y0). a is the radius of the cylinder.
% d is the length of the cylinder. H is the height of the top of the cylinder.
% G is Newtons gravitational constant. sigma is the terrain density.
global G sigma H X0 Y0 a d

r=sqrt( (x-X0).^2 + (y-Y0).^2);
D=sqrt( (H+z)^2 + (a+r)^2 );
F=sqrt( (H+d+z)^2 + (a+r)^2 );
A=(4*a*r)./(D^2);
B=(4*a*r)./(F^2);
[KA,EA]=ellipke(A);
[KB,EB]=ellipke(B);
if r~=0
    h=G*sigma*1e9*( (y-Y0)/(r^2) )*D*( (2-A)*KA-2*EA )-F*( (2-B)*KB-2*EB);
 else
    h=0;
end;
```

The der-x-Txz.m routine

```
function hx=der_x_Txz(x,y,z);
% hx=der_x_Txz(x,y,z);
%
% This function returns the derivative with respect to x
% of the gravity gradient response g_{xz}

dx=0.001;
hx=(Tx(x+dx,y,z)-Tx(x,y,z))/dx;
```

The der-y-Tyz.m routine

```
function hy=der_y_Tyz(x,y,z);
% hy=der_y_Tyz(x,y,z);
%
% This function returns the derivative with respect to y
% of the gravity gradient response g_{yz}

dy=0.001;
hy=(Ty(x,y+dy,z)-Ty(x,y,z))/dy;
```

The Gzzz.m routine

```
function hz=Gzzz(x0)
% hz=Gzzz(x0);
%
% This function returns the g_{zzz} response given the
% g_{xz} and g_{yz} responses.

% The center of the cylinder lies at (X0,Y0). a is the radius of the cylinder.
% d is the length of the cylinder. H is the height of the top of the cylinder.
% G is Newtons gravitational constant. sigma is the terrain density.
global G sigma H X0 Y0 a d ;

hz=der_x_Txz(x,y,z)+der_y_Tyz(x,y,z);
```

The cgenv.m routine

```
function cout=cgenv(a,N);
% cout=cgenv(a,N);
%
% This function generates the contour model for the cylinder.
% The input data  are a, the radius of the cylinder and N,
% the number of levels in the contour elevation. The routine
% returns the contour matrix c for the cylinder.

delta=pi/36;
thetao=pi/4;
theta=(thetao: delta:(2*pi + thetao + 0.1*delta ));
nps=max(size(theta));
```

```
dz=2*(R-0.001)/(N-1);
z=-(R-0.001):dz:(R-0.001);
c=zeros(2,N*7);
for i=1:N;
    p=(i-1)*(nps+1);
    c(1,p+1)=z(i);
    c(2,p+1)=nps;
    in=(p+2):(p+nps+1);
    rad=5;
    c(1,in)=rad*cos(theta);
    c(2,in)=rad*sin(theta);
end%for
cout=c;
```

The potgg.m routine

```
function g=potgg(xdata,ydata,zdata,xobs,dz);
% g=potgg(xdata,ydata,zdata,xobs,dz);
%
% This function evaluates the gravity gradient response from the polygon
% with vertices stored in the vectors xdata, ydata and zdata. The response
% is computed at observation point xobs.

M=max(size(xdata));
smm=0;
A=0;
z2=-(zdata(1)-abs(xobs(3)));
z1=-(-z2+dz);
for i=1:(M-1),
    if~((xdata(i)==xdata(i+1))&(ydata(i)==ydata(i+1)));
      x(1)=(xdata(i)-xobs(1));
      x(2)=(xdata(i+1)-xobs(1));
      y(1)=(ydata(i)-xobs(2));
      y(2)=(ydata(i+1)-xobs(2));
      dx=x(2)-x(1);
      dy=y(2)-y(1);
      ds=sqrt(dx^2+dy^2);
      p=(x(1)*y(2)-x(2)*y(1))/ds;
      r1=sqrt(x(1)^2+ y(1)^2);
      r2=sqrt(x(2)^2+y(2)^2);
      s=dx/ds;
      c=dy/ds;
```

```
      d1=x(1)*s+y(1)*c;
      d2=x(2)*s+y(2)*c;
      R11=sqrt(z1^2+r1^2);
      R21=sqrt(z1^2+r2^2);
      R12=sqrt(z2^2+r1^2);
      R22=sqrt(z2^2+r2^2);
      nmr1=d1*p*r1*r1/R11; nmr2=d2*p*r2*r2/R21;
      dmr1=p*p*R11*R11+d1*d1*z1*z1; dmr2=p*p*R21*R21+d2*d2*z1*z1;
      phi=-(nmr1/dmr1-nmr2/dmr2);
      nmr1=d1*p*r1*r1/R12; nmr2=d2*p*r2*r2/R22;
      dmr1=p*p*R12*R12+d1*d1*z2*z2; dmr2=p*p*R22*R22+d2*d2*z2*z2;
      phi=phi+nmr1/dmr1-nmr2/dmr2;
      smm=smm+phi;
    end;% if
end;% for i
g=smm;
```

The pggzzz.m routine

```
function y=pgzzz(c,xobs,dz);
% y=pgzzz(c,xobs,dz);
%
% This function is used to calculate the g_{zzz}
% response sourced by the terrain described by the
% contour matrix c. The response is computed at the
% observation point with coordinates xobs. Each layer in
% the contour model (i.e. each polygon) has thickness dz.

csize=size(c);
limit=csize(2);
hh=[];
elevs=[];
i=1;
pp=0;
zold=0;
while(i < limit)
    z_level=c(1,i);
    npoints=c(2,i);
    nexti=i+npoints+1;
    pp=pp+1;
    xdata=c(1,i+1:i+npoints);
    ydata=c(2,i+1:i+npoints);
    zdata=z_level*ones(1, npoints);
    % compute the response for one polygon
```

```
    hh(pp)= potgg(xdata,ydata,zdata,xobs,dz);
    i=nexti;
 end %while
% sum to obtain total response
g=sum(hh);
gamma=6.67/(1e11);
Grho=200*gamma;
g=Grho*g*1e9;
```

The runcgggz.m routine

```
% runcgggz.m
%
% This function compares the gravity gradient responses
% computed numerically to the exact responses. The contour
% model for the cylinder, the numerical and analytic
% responses and the error in the numerical response are
% plotted.

% initialization
clear all;
close all;

% The center of the cylinder lies at (X0,Y0). a is the radius of the cylinder.
% d is the length of the cylinder. H is the height of the top of the cylinder.
% G is Newtons gravitational constant. sigma is the terrain density.
global G sigma H X0 Y0 a d ;

G=6.67/100000000000;
sigma=200;
a=5;
H=-5;
X0=0
Y0=0
R=5
N=10
c=cgenv(R,N);
dz=2*(R-0.001)/(N-1);
z=20
d=2*z/(N-1)

subplot(2,2,1);
% plot the contour model
pltc(c);
```

```
title('contour Model');
xlabel('position in meter');
ylabel('position in meter');
zlabel('position in meter');

% da sets the observation grid
da=11;
hd=(da+1)/2
for i= 1:da
    x=(i-hd)
    for j=1:da
        y=(j-hd)
        xobs=[(i-hd),(j-hd),20];
        z=xobs(3)
        % numerical response
        grav1(i,j)=pggzzz(c,xobs,dz)
        % exact response
        gzzz2(i,j)=Gzzz(x,y,z);
    end%for j
    x(i)=xobs(1);
end%for i

% error in numerical response
gzzz=(grav1-gzzz2);

subplot(2, 2, 2);
% plot exact response
mesh(grav1);
title('Analytic solution');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');

subplot(2, 2, 3);
mesh(gzzz2);
% plot numerical response
title('Numerical solution');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');

subplot(2, 2, 4);
% plot the error in numerical response
mesh(gzzz);
title('Error');
```

```
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');
```

The cvgg.m routine

```
function yout=cvgg(xo);

% yout=cvgg(xo);
%
% This function returns the exact g_{zzz} response sourced by a shere.
% The input parameter is the observation point xo.
% The radius R of the sphere is set internally.

% set radius of sphere
R=2

x=xo(1);
y=xo(2);
z=xo(3);
gamma=6.67/(1e11);
Grho=300*gamma;
r=sqrt(x*x+y*y+z*z);
gzzz=4*pi*R*R*R*((5*z*z*z-3*z*r*r)/(r^7));
% normalize response
yout=-Grho*gzzz*(1e9);
```

The genvgg.m routine

```
function cout=genvgg(R,N);

% cout=genvgg(R,N);
%
% This function generates the contour model for a sphere.
% The input data are R, the radius of the sphere, and N,
% the number of elevations in the contour model. The
% routine returns a contour matrix c.

delta=pi/36;
thetao=pi/4;
```

```
theta=(thetao: delta:(2*pi + thetao + 0.1*delta ));
nps=max(size(theta));
dz=2*(R-0.001)/(N-1);
z=-(R-0.001):dz:(R-0.001);
c=zeros(2,N*6);
for i=1:N;
    p=(i-1)*(nps+1);
    c(1,p+1)=z(i);
    c(2,p+1)=nps;
    in=(p+2):(p+nps+1);
    rad=sqrt(R*R-z(i)*z(i));
    c(1,in)=rad*cos(theta);
    c(2,in)=rad*sin(theta);
end%for
cout=c;
```

The pgzzz.m routine

```
function g=pgzzz(c,xobs,dz);
% g=pgzzz(c,xobs,dz);
%
% This function computes the g_{zzz} response for the terrain modelled by
% the contour matrix c. The routine calls routine potgg to compute the
% g_{zzz} response sourced by a single polygon.

csize=size(c);
% limit stores the number of columns in the contour matrix
limit=csize(2);

% initialize variables
hh=[];
elevs=[];
i=1;
pp=0;
zold=0;
while(i < limit)
    z_level=c(1,i);
    npoints=c(2,i);
    nexti=i+npoints+1;
    pp=pp+1;
    xdata=c(1,i+1:i+npoints);
    ydata=c(2,i+1:i+npoints);
    zdata=z_level*ones(1, npoints);
    % compute response from a single polygon.
```

```
    hh(pp)= potggg(xdata,ydata,zdata,xobs,dz);
    i=nexti;
end %while
% obtain total response
g=sum(hh);

gamma=6.67/(1e11);
Grho=300*gamma;
% normalize total response
g=Grho*g*1e9;
```

The potggg.m routine

```
function g=potggg(xdata,ydata,zdata,xobs,dz);
% g=potggg(xdata,ydata,zdata,xobs,dz);
%
% This function evaluates the g_{zzz} response, at the observation
% point xobs, for a single polygon. The vertices of the polygon
% are stored in the vectors xdata, ydata and zdata. The thickness
% of the polygon is given by dz. The routine implements a second
% derivative (with respect to z) of the formula derived by Plouff.

M=max(size(xdata));
smm=0;
A=0;
z2=-(zdata(1)-abs(xobs(3)));
z1=-(-z2+dz);
for i=1:(M-1),
    if~((xdata(i)==xdata(i+1))&(ydata(i)==ydata(i+1)));
      x(1)=(xdata(i)-xobs(1));
      x(2)=(xdata(i+1)-xobs(1));
      y(1)=(ydata(i)-xobs(2));
      y(2)=(ydata(i+1)-xobs(2));
      dx=x(2)-x(1);
      dy=y(2)-y(1);
      ds=sqrt(dx^2+dy^2);
      p=(x(1)*y(2)-x(2)*y(1))/ds;
      r1=sqrt(x(1)^2+ y(1)^2);
      r2=sqrt(x(2)^2+y(2)^2);
      s=dx/ds;
      c=dy/ds;
      d1=x(1)*s+y(1)*c;
      d2=x(2)*s+y(2)*c;
```

```
      R11=sqrt(z1^2+r1^2);
      R21=sqrt(z1^2+r2^2);
      R12=sqrt(z2^2+r1^2);
      R22=sqrt(z2^2+r2^2);
      nmr1=d1*p*r1*r1/R11; nmr2=d2*p*r2*r2/R21;
      dmr1=p*p*R11*R11+d1*d1*z1*z1; dmr2=p*p*R21*R21+d2*d2*z1*z1;
      phi=-(nmr1/dmr1-nmr2/dmr2);
      nmr1=d1*p*r1*r1/R12; nmr2=d2*p*r2*r2/R22;
      dmr1=p*p*R12*R12+d1*d1*z2*z2; dmr2=p*p*R22*R22+d2*d2*z2*z2;
      phi=phi+nmr1/dmr1-nmr2/dmr2;
      smm=smm+phi;
    end; % if
end;% for i
g=smm;
```

The runvgg.m routine

```
% runvgg.m
%
% This function compares the g_{zzz} responses from a sphere
% computed numerically to the exact responses. The contour
% model for the sphere, the numerical and analytic
% responses and the error in the numerical response are
% plotted.

% initialization
clear all;
close all;

% the radius od the sphere
R=2;
% the number of elevations in the contour model
N=20;
% generate contour model
c=genvgg(R,N);
dz=-2*(R-0.001)/(N-1);

subplot(2,2,1);
% plot contour model
pltc(c)
title('Contour Model');
xlabel('position (Meters)');
ylabel('position (Meter)');
```

```
zlabel('position (Meter)');

% set observation grid
d=11; hd=(d+1)/2;
for i=1:d,
    for j=1:d,
        xobs=[(i-hd),(j-hd),20];
        % numerical response
        grav1(i,j)=pgzzz(c,xobs,dz);
        % exact response
        grav2(i,j)=cvgg(xobs);
    end; %for j
    x(i)=xobs(1);
end;% for i
% compute the error
grav=(grav1-grav2);

subplot(2, 2, 2);
% plot numerical response
mesh(x,x,grav1)
title('Method of lamina');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');

subplot(2, 2, 3);
% plot exact response
mesh(x,x,grav2);
title('Exact Numerical solution');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');

subplot(2, 2, 4);
% plot error
mesh(x,x,grav);
title('Error');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eovos');
```

The bigcub.m routine

```
% bigcub.m
%
% This function is use to compute the gravity response
% from a cube and a Gaussian. Together, the two model a
% hill on a flat landscape. This routine is used to
% estimate how much of the distant terrain needs to be
% convolved to obtain an accurate answer.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Build contour model for terrain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% the number of elevations in the Gaussian contour model
j=20

% the grid of points used to produce contour model
x=-2:.2:2;
y=-2:.2:2;
[x,y]=meshgrid(x,y)

% the Gaussian used to model the hill
k=exp(-sqrt(x.^2+y.^2))

subplot(2, 2, 1)
% plot hill
c2=contour3(x,y,k+1,j);
hold on
view(3);
title('Gaussian contour')
xlabel('position in meter');
ylabel(' position in meter');
zlabel('position in meter');
hold off
grid off

% vertices of the cube
vertice_list=[0 0 0;1 0 0; 1 1 0; 0 1 0;0 0 1 ; 1 0 1 ;1 1 1 ; 0 1 1]
% faces of the cube
faces=[1 2  6 5; 2 3 7 6 ; 3 4 8 7; 4 1 5 8; 1 2 3 4; 5 6 7 8]
[X,Y]=meshgrid(vertice_list,faces)
c3=contourc(X,Y)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute response from terrain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
pp1=([ c2])
d=11; hd=(d+1)/2;
zc=-2
N=20
dz=-2*zc/(N-1);
zc=-2;
for i=1:d,
    for j=1:d,
       xobs=[6.7*(i-hd),6.7*(j-hd),-40];
       % compute response
       grav1(i,j)=pgzzD(pp1,xobs,dz);
    end;%for j
    x(i)=xobs(1);
end;

subplot(2, 2, 2);
% plot response
mesh(grav1)
title('Gravitational Response');
xlabel('position in meter');
ylabel(' position in meter');
zlabel('Response in Eotvos');
```

The function Gvd.m

```
function h=Gvd(theta,x1,y1,z1,j);
% h=Gvd(theta,x1,y1,z1,j);
%
% This function returns the integrand of the kernel of the RKHS. To obtain
% the kernel K one needs to integrate over theta.

j=sqrt(-1);
h=log(-(j*cos(theta).*x1+j*sin(theta).*y1-z1)).*[(-(j*cos(theta).*x1+...
  ...j*sin(theta).*y1-z1))+z1]-(-(j*cos(theta).*x1+j*sin(theta).*y1-z1));
```

The function testfilter.m

```
function y=testfilter(d);
```

```
% function y=testfilter(d);
%
% There are d^2 points in a grid around a Gaussian hill.
% This routine displays the measured noisy response and
% the result of interpolation.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the landscape
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=-20:0.25:20;
X=x'*ones(size(x));
Y=X';
c=contourc(x,x,2*exp(0.4*(-X.^2-Y.^2)),10);
dz=0.2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the landscape contour model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,2,1);
pltc(c);
title('Contour Model');
xlabel('position (Meters)');
%ylabel('position (Meter)');
zlabel('position (Meter)');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the observation grid and the grid to be used for
% interpolation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hd=(d+1)/2;
Xobs=(10*(1:d)-10*hd);
for i=1:d,
   for j=1:d,
      obsgridx(i+(j-1)*d)=10*(i-hd)+5*randn; % random smaple
      obsgridy(i+(j-1)*d)=10*(j-hd)+5*randn; % positions
      obsgridz(i+(j-1)*d)=40;
      intgridx(i+(j-1)*d)=10*(i-hd);
      intgridy(i+(j-1)*d)=10*(j-hd);
      intgridz(i+(j-1)*d)=40;
   end; %for j
end;% for i

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Compute the gzz and gzzz responses from the terrain
% grav stores gzz
% ggrav stores gzzz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for p=1:(d^2),
   xobs=[obsgridx(p),obsgridy(p),obsgridz(p)];
   grav(p)=pgzz(c,xobs,dz);
   ggrav(p)=pgzzz(c,xobs,dz);
end; %for j

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Turn the (noise free) gravity gradient and observation grids
% into matrices that can be plotted using mesh
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:d,
   for j=1:d,
      Mgrav(i,j)=grav(i+(j-1)*d);
      Xobs(i,j)=obsgridx(i+(j-1)*d);
      Yobs(i,j)=obsgridy(i+(j-1)*d);
      iXobs(i,j)=intgridx(i+(j-1)*d);
      iYobs(i,j)=intgridy(i+(j-1)*d);
   end; %for j
end;% for i

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the noise free gravity response
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(2, 2, 2);
mesh(Xobs,Yobs,Mgrav)
title('Method of lamina');
xlabel('position in meter');
zlabel('Response in Eotvos');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute the noise in the signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

noise=0*5*rand(size(grav));
ngrav=grav+ggrav.*noise;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Turn the noisy gravity gradient into a matrix that can
```

```
% be plotted using mesh
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:d,
   for j=1:d,
      Ngrav(i,j)=ngrav(i+(j-1)*d);
   end; %for j
end;% for i

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the noisy gravity response
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(2, 2, 3);
mesh(Xobs,Yobs,Ngrav)
title('Method of lamina');
xlabel('position in meter');
zlabel('Response in Eotvos');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute kernel for obs grid and int grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for I=1:(d^2),
   for J=1:(d^2),
      xv(1)=obsgridx(I)-intgridx(J);
      xv(2)=obsgridy(I)-intgridy(J);
      xv(3)=40;
      K(I,J)=quad8('Gvd',0,2*pi,[],[],xv);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Invert kernel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

invK=inv(K);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the new grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

d2=(d-1)*2+1; hd=(d2+1)/2;
for i=1:d2,
   for j=1:d2,
```

```
      pobsgridx(i+(j-1)*d2)=5*(i-hd);
      pobsgridy(i+(j-1)*d2)=5*(j-hd);
      pobsgridz(i+(j-1)*d2)=40;
   end; %for j
end;% for i

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute kernel for int grid and prediction grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for I=1:(d2^2),
   for J=1:(d^2),
      xv(1)=-intgridx(J)+pobsgridx(I);
      xv(2)=-intgridy(J)+pobsgridy(I);
      xv(3)=40;
      K(I,J)=quad8('Gvd',0,2*pi,[],[],xv);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute exact gravity response for prediction grid. This is
% used to compare against the interpolant.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for I=1:(d2^2),
   xv(1)=pobsgridx(I);
   xv(2)=pobsgridy(I);
   xv(3)=40;
   truegrav(I)=pgzz(c,xv,dz);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute Interpolated Spectrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

KKinv=K*invK;
%pred=KKinv*ngrav';
pred=KKinv*grav';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Turn the (noise free) gravity gradient and interpolant
% into matrices that can be plotted using mesh
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for Ii=1:d2,
```

```
   for Ij=1:d2,
      Pred(Ii,Ij)=pred(Ii+(Ij-1)*d2);
      Tgrav(Ii,Ij)=truegrav(Ii+(Ij-1)*d2);
      Pobsgridx(Ii,Ij)=pobsgridx(Ii+(Ij-1)*d2);
      Pobsgridy(Ii,Ij)=pobsgridy(Ii+(Ij-1)*d2);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute linear interpolant
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:(d-1),
   for j=1:(d-1),
      %Igrav(2*i-1,2*j-1)=Ngrav(i,j);
      %Igrav(2*i-1,2*j-1)=Mgrav(i,j);
      x1=Xobs(i,j);     y1=Yobs(i,j);
      x2=Xobs(i+1,j);   y2=Yobs(i+1,j);
      x3=Xobs(i,j+1);   y3=Yobs(i,j+1);
      nx1=iXobs(i,j);   ny1=iYobs(i,j);
      M=[x1, y1, 1; x2, y2, 1; x3, y3, 1];
      z=[Mgrav(i,j); Mgrav(i+1,j); Mgrav(i,j+1)];
      coeffs=inv(M)*z;
      Igrav(2*i-1,2*j-1)=coeffs(1)*nx1+coeffs(2)*ny1+coeffs(3);
   end;
end;

for i=1:(d-1), % j=d
   x1=Xobs(i,d);     y1=Yobs(i,d);
   x2=Xobs(i+1,d);   y2=Yobs(i+1,d);
   x3=Xobs(i,d-1);   y3=Yobs(i,d-1);
   nx1=iXobs(i,d);   ny1=iYobs(i,d);
   M=[x1, y1, 1; x2, y2, 1; x3, y3, 1];
   z=[Mgrav(i,d); Mgrav(i+1,d); Mgrav(i,d-1)];
   coeffs=inv(M)*z;
   Igrav(2*i-1,2*d-1)=coeffs(1)*nx1+coeffs(2)*ny1+coeffs(3);
end;

for j=1:(d-1), % i=d
   x1=Xobs(d,j);     y1=Yobs(d,j);
   x2=Xobs(d-1,j);   y2=Yobs(d-1,j);
   x3=Xobs(d,j+1);   y3=Yobs(d,j+1);
   nx1=iXobs(d,j);   ny1=iYobs(d,j);
   M=[x1, y1, 1; x2, y2, 1; x3, y3, 1];
   z=[Mgrav(d,j); Mgrav(d-1,j); Mgrav(d,j+1)];
```

```
   coeffs=inv(M)*z;
   Igrav(2*d-1,2*j-1)=coeffs(1)*nx1+coeffs(2)*ny1+coeffs(3);
end;

% i=j=d
x1=Xobs(d,d);     y1=Yobs(d,d);
x2=Xobs(d-1,d);   y2=Yobs(d-1,d);
x3=Xobs(d,d-1);   y3=Yobs(d,d-1);
nx1=iXobs(d,d);   ny1=iYobs(d,d);
M=[x1, y1, 1; x2, y2, 1; x3, y3, 1];
z=[Mgrav(d,d); Mgrav(d-1,d); Mgrav(d,d-1)];
coeffs=inv(M)*z;
Igrav(2*d-1,2*d-1)=coeffs(1)*nx1+coeffs(2)*ny1+coeffs(3);

for i=1:d,
   for j=1:(d-1),
      Igrav(2*i-1,2*j)=(Igrav(2*i-1,2*j-1)+Igrav(2*i-1,2*j+1))/2;
   end;
end;

for j=1:d,
   for i=1:(d-1),
      Igrav(2*i,2*j-1)=(Igrav(2*i-1,2*j-1)+Igrav(2*i+1,2*j-1))/2;
   end;
end;

for i=1:(d-1),
   for j=1:(d-1),
      Igrav(2*i,2*j)=(Igrav(2*i,2*j-1)+Igrav(2*i,2*j+1))/2;
   end;
end;

% Noise in our interpolant
y1=sqrt(mean(mean((Pred-Tgrav).^2)));

% Noise in linear interpolant
y2=sqrt(mean(mean((Tgrav-Igrav).^2)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the interpolant
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(2,2,4)
mesh(Pobsgridx,Pobsgridy,real(Pred))
```

```
y=[y1; y2];
```

Bibliography

[1] D. Plouff, Gravity and magnetic field of polygonal prisms and application to magnetic terrain corrections, *Geophysics*, 41:727–41, 1976.

[2] L.L. Nettleton, Gravity and Magnetics in oil prospecting, McGraw-Hill international series in earth and planetary science, 1976.

[3] M. Talwani and M. Ewing, Rapid Computation of Gravitational Attraction of Three Dimensional Bodies of Arbitrary Shape, *Geophysics*, 25:205–225, 1960.

[4] Rasmussen L. Pedersen, The Gradient tensor of potential field anomalies: some implications on data collection and data processing of maps, *Geophysics*, 55, 1990.

[5] R.J. Blakely, Potential Theory in Gravity and Magnetic Applications, Cambridge University Press, 1996.

[6] A. Veryaskin and W. McRae On the combined gravity gradient components modeling for applied geophysics, *arXiv:0707.2821*.

[7] Gregory R. Wessel, Sulfur Resources: Chapter 86 - Part 2 Commodities and Uses Industrial Minerals and Rocks 6th Edition

[8] A. P. Prudnikov, Yu. A. Brychkov, and O. I. Marichev, Integrals and Series, vols 1 and 2, Oxford:Gordon and Breach 1986.

[9] K.F Riley, M.P Hobson, and S.J Bence, Mathematical Methods for Physics and Engineering, Press syndicate of the university of Cambridge, 2003.

[10] P.M. Morse and H. Feshbach, Methods of Theoretical Physics, McGraw Hill, New York, 1953.

[11] G. Chen and R. J. P. de Figueiredo, A Unified Approach to Optimal Image Interpolation Problems based on Linear Partial Differential Equation Models, *IEEE Trans. Imag. Proc.* 2, 519–528, 1993.

[12] J. Maltz, R. de Mello Koch and A. J. Willis, Reproducing Kernel Hilbert Space Method for Optimal Interpolation of Potential Field Data, *IEEE Trans. Imag. Proc.* 7, 1724, 1998.

[13] Nachman Aronszajn, Theory of Reproducing Kernels, Transactions of the American Mathematical Society, 68, 337404, (1950).

[14] Kearey Brooks Hill, An Interduction to Geophysical Exploration, Iowa state University press, 2002.