Topography Estimation Using Particle Swarm Optimization

Yelbir Kazhykarim
The University of Texas Rio Grande Valley

TOPOGRAPHY ESTIMATION USING PARTICLE SWARM OPTIMIZATION

A Thesis

by

YELBIR KAZHYKARIM

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2020

Major Subject: Physics

TOPOGRAPHY ESTIMATION USING PARTICLE SWARM OPTIMIZATION

A Thesis
by
YELBIR KAZHYKARIM

COMMITTEE MEMBERS

Dr. Soumya Mohanty
Chair of Committee

Dr. Soma Mukherjee
Committee Member

Dr. Teviet Creighton
Committee Member

May 2020

ABSTRACT

Kazhykarim,Yelbir, Topography estimation using Particle Swarm Optimization. Master of Science (MS), May, 2020, 42 pp., 21 figures, 7 references.

The Earth's Vertical Gravity Gradient (VGG) can be used to map seafloor topography but presents a challenging inverse problem. A promising approach is forward modeling, in which one searches over a set of candidate topographies and selects the one whose predicted VGG best fits the observed one. The main bottleneck here is solving the associated high-dimensional and non-linear optimization problem. Yang et al (2018) demonstrated a method in which the topography is parametrized by heights of mass elements on a rectangular grid and the $\approx 10^4$ dimensional optimization problem is tackled with simulated annealing (SA). We propose a computationally much cheaper method, using a stochastic optimization method known as Particle Swarm Optimization (PSO) and representing the topography as a linear combination of Radial Basis Functions (RBFs). First results, obtained without any tuning, show that the MATLAB code achieves an RMS error of 650 m with 500 RBFs (1500 parameters) and a 30 min run time. This is comparable to the error of 300 m from the much more expensive SA method that takes hours on a super-computer. Improvements to our method are likely to result in state of the art performance levels.

DEDICATION

I am blessed with having the best parents in the world and a brother who also happens to be my best friend and always supports me in all my endeavors. Without them I would have never succeeded - and to them I dedicate this work.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Mohanty, for his continuous support, deep insight and calmness while helping in my constant struggle with bugs. I wouldn't have chosen anyone else by my side, even if the bugs were from Starship Troopers.

I would also like to express my gratitude to Dr. Junjun Yang for suggesting the problem to my advisor and kindly providing the data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

SEAFLOOR TOPOGRAPHY AND GRAVITY GRADIENTS

1.1 Background

Our knowledge of the seafloor topography is worse than that of the surface of Mars [1],[2]. Currently only 18% of the seafloor is mapped, and that is after 60 years of ship sounding [1], [3]. To demonstrate how poorly the ocean floor is mapped, Smith et al [2] indicate that the interstate highway system in USA has coverage density comparable to that of the bathymetry surveys, as illustrated in Figure 1.1.

This is simply a result of the vastness of the oceans - it has been estimated that doing it directly using ship sounding would require 900 ship-years, which gives around 200 more years of work if current speed is extrapolated [4].

In the meantime, the infamous airplane accident of the Malaysia airlines flight MH370 in 2014 has clearly demonstrated the practical consequences of having limited maps: the aircraft went missing in the Indian Ocean and it took several weeks to locate it [5].

Thus there is a need for better understanding of the indirect methods of inferring topography. The main approach used in this field is based on estimating topography from gravity measurements. The first time these ideas were mentioned in the literature was in 18th century, with proposals at estimating depth first made in 19th century [2]. However, at that time it was nothing but curiosity due to the technological limitations of the era.

Such methods are known since 1970's. The method developed by Parker [6], was, for instance, based on a series expansion of the Fourier image of the topography. Until recently, the use of the series, however, has been limited by the available computational resources. Thus, Smith and Sandwell, for example, who used Parker's method in the 90's [3], could only use the first term of

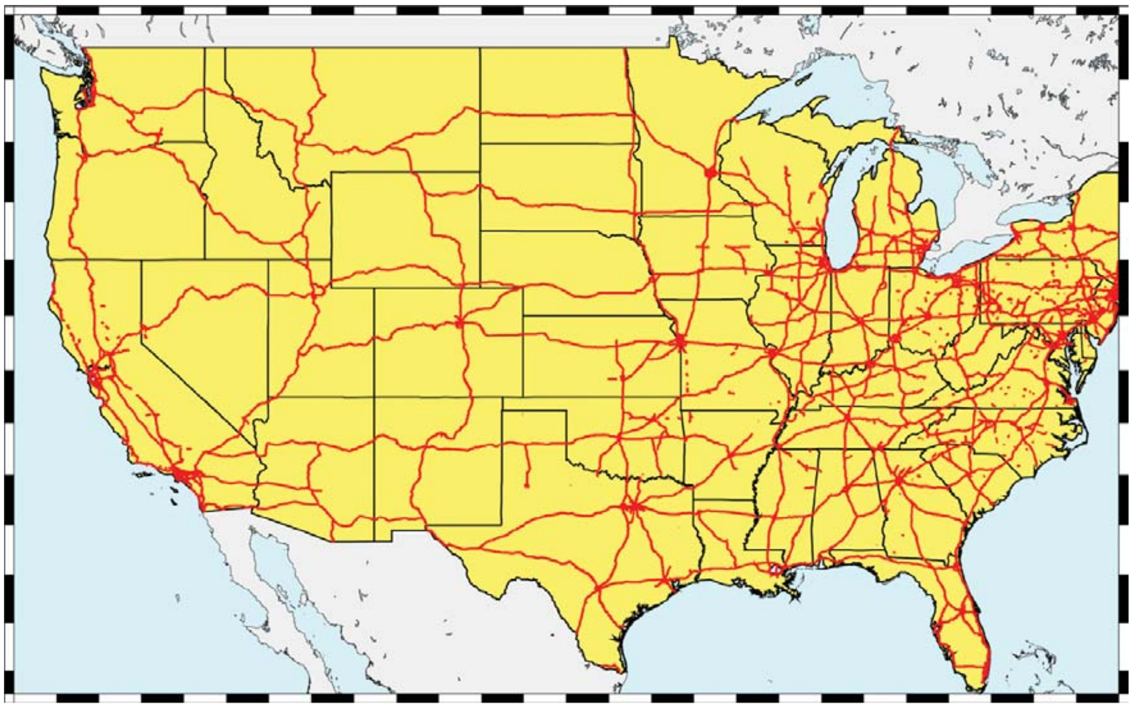
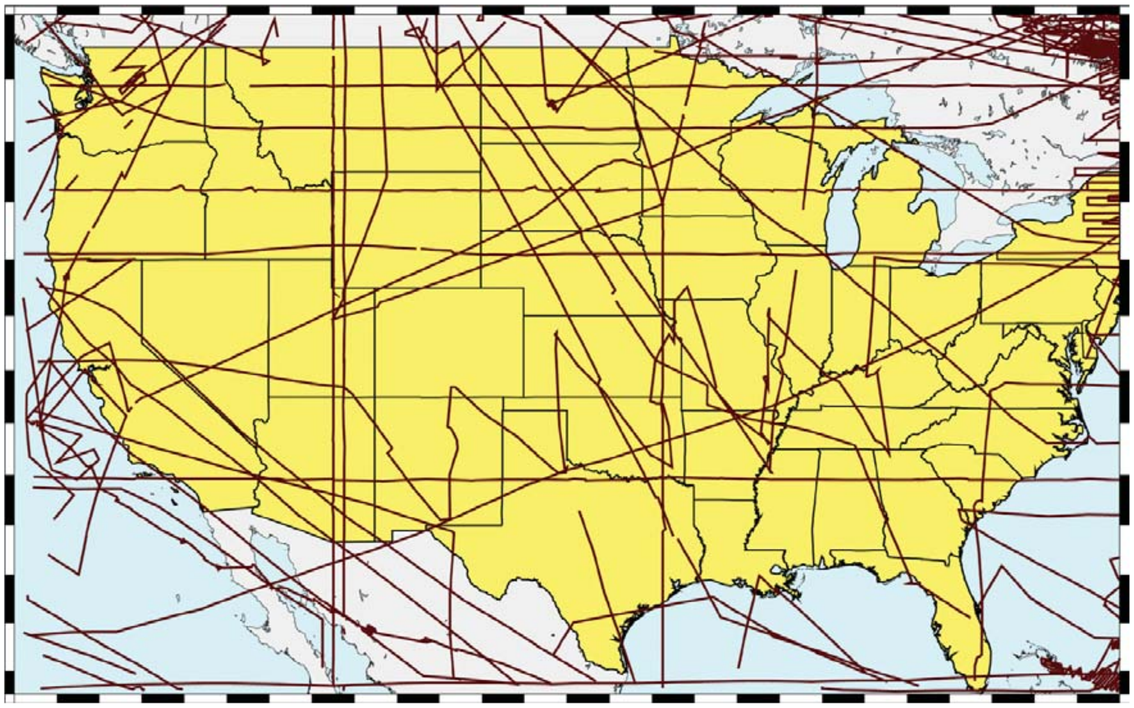



Figure 1.1: Interstate highway system (top) and bathymetry survey lines (botton). Both are shown to the same scale. Reprinted from: W. H. F. Smith, and D. T. Sandwell. "Conventional bathymetry, bathymetry from space, and geodetic altimetry." *Oceanography* - 17 (vol. 1) (2004): 8-23.

the series, i.e. the linear approximation.

1.2 Inverse problem

Having established the need for indirect methods of estimating topography, I hereby present the two methods based on the use of Vertical Gravity Gradients (VGG) and solving either the *inverse* or the *forward* problems.

First, the definition of VGG should be presented. VGG, typically denoted by Γ_{zz}, is the vertical component of the gradient of the vertical component of the acceleration of gravity, i.e. $\Gamma_{zz} = \frac{\partial g_z}{\partial z}$, assuming the z axis to be vertical. In general, VGG is a function of all three spatial coordinates. In this work, however, only VGG values measured at the sea level are to be employed. We use a planar approximation and hence are left with VGG as the function of x and y: $\Gamma_{zz} = \Gamma_{zz}(x,y)$.

There is a relationship between Fourier images of topography and of VGG, first obtained by Parker in 1973 [6] that plays a key role in our method. To derive it, we first establish the notation. Following Parker, we introduce a Cartesian coordinate system with $\hat{\mathbf{z}}$ pointing vertically upwards, so that a position vector $\mathbf{r}$ = $\mathbf{r}$(x,y,z). Next, again following Parker, we introduce a somewhat idiosyncratic notation in which the x-y projection vector of $\mathbf{r}$ is denoted by $\vec{r}$. As demonstrated in Figure 1.2 the following relationship holds true:

$$\vec{r} = \mathbf{r} - \hat{\mathbf{z}}(\hat{\mathbf{z}} \cdot \mathbf{r}) \tag{1.1}$$

Now, the two dimensional Fourier transform is defined as:

$$\mathscr{F}[f(\vec{r})] = \iint_X dS\, f(\vec{r})\; e^{i\vec{k}\cdot\vec{r}}, \tag{1.2}$$

where $\vec{k}$ is the wave vector and X represents the x-y plane. Note that $\vec{k}$ is constrained to the x-y plane.

Before we can find the Fourier image of the gravitational potential, we should make several

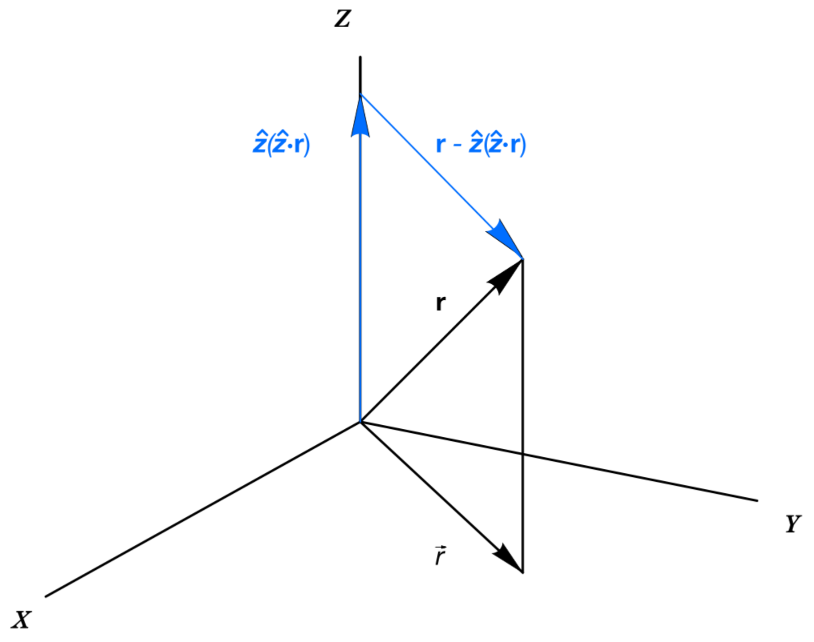

Z
$\hat{z}(\hat{z}\cdot\mathbf{r})$
$\mathbf{r} - \hat{z}(\hat{z}\cdot\mathbf{r})$
$\mathbf{r}$
Y
$\vec{r}$
X

Figure 1.2: Coordinate System

assumptions. First, we consider the gravitational potential of a layer of material and set up our coordinate system so that z = 0 plane corresponds to the lower boundary of the material. The layer should vanish outside of a circle with radius: if $r > R$, then h($\vec{r}$) = 0. This assumption guarantees the absence of convergence problems [6]. Second, h is assumed to be a 'well-behaved', bounded and integrable function. Density of the source on the seafloor is assumed constant. Hence, under these reasonable assumptions, it can be stated that the gravitational potential U of the layer at a position $\mathbf{r}_0$ is given by:

$$U(\mathbf{r}_0) = G\rho \iiint_V dV \,/|\mathbf{r}_0 - \mathbf{r}| \tag{1.3}$$

Or, in a little more detail:

$$U(\mathbf{r}_0) = G\rho \int_D dS \int_0^{h(\mathbf{r})} dz/|\mathbf{r}_0 - \mathbf{r}|, \tag{1.4}$$

where G is the gravitational constant and ρ is constant density. Now, if the observation point is restricted by the plane $z = z_0$, where z_0 is assumed to be above all topography and in this work is chosen to correspond to the sea level, then U becomes a function only of $\vec{r}_0$. Therefore, taking the Fourier transform of (1.4), one obtains:

$$\mathscr{F}[U(\vec{r}_0)] = \int_X dS_0 U(\vec{r}_0)\, e^{i\vec{k}\cdot\vec{r}_0} = G\rho \int_X dS_0 \int_D dS\, e^{i\vec{k}\cdot\vec{r}_0} \int_0^{h(\mathbf{r})} dz/|\mathbf{r}_0 - \mathbf{r}| = \\ G\rho \int_D dS \int_0^{h(\mathbf{r})} dz \int_X dS_0\, e^{i\vec{k}\cdot\vec{r}_0}/|\mathbf{r}_0 - \mathbf{r}|, \tag{1.5}$$

where in the last step we exchange the order of integration for convenience - the last integral in (1.5) can now be evaluated in polar coordinates. Using $|\vec{k}| = k$, the result is:

$$\mathscr{F}[U(\vec{r}_0)] = G\rho \int_D dS \int_0^{h(\mathbf{r})} dz\, [2\pi\, e^{i\vec{k}\cdot\mathbf{r} - k(z_0 - z)}]/k \tag{1.6}$$

Finally, after evaluating the z integral one obtains:

$$\mathscr{F}[U(\vec{r}_0)] = 2\pi G\rho \int_D dS\, (e^{i\vec{k}\cdot\vec{r} - kz_0})(e^{kh(\vec{r})} - 1)/k^2 \tag{1.7}$$

Now, expanding the second exponential in (1.7) in Taylor series, then exchanging the order of summation and integration, we finally get:

$$\mathscr{F}[U(\vec{r}_0)] = 2\pi G\rho e^{-kz_0} \sum_{n=1}^{\infty} \frac{k^{n-2}}{n!} \mathscr{F}[h^n(\vec{r})] \tag{1.8}$$

We need to find the VGG, however, not the potential. Using the inverse Fourier transform, the potential can be written as:

$$U(\vec{r}_0) = \frac{1}{4\pi^2} \iint d^2k\, \bar{U}(\vec{k}) e^{-k\hat{z}\cdot\mathbf{r_0} - i\vec{k}\cdot\mathbf{r_0}} \tag{1.9}$$

Therefore one obtains:

$$\mathscr{F}[U(\mathbf{r_0})] = \bar{U}(\vec{k}) e^{-k\hat{z}_0\cdot\mathbf{r_0}} \tag{1.10}$$

Now, VGG is defined as $\Gamma_{zz} = \frac{\partial g_z}{\partial z} = \frac{\partial^2 U(\mathbf{r_0})}{\partial z^2}$. Therefore, by differentiating one obtains:

$$\mathscr{F}[\Gamma_{zz}] = k^2 \mathscr{F}[U(\mathbf{r_0})] \tag{1.11}$$

Hence, one finally obtains an expression known as Parker's series:

$$\mathscr{F}(\Gamma_{zz}) = 2\pi G \Delta\rho \exp^{-2\pi \bar{f} z_0} \sum_{n=1}^{n=\infty} \frac{(2\pi \bar{f})^n}{n!} \mathscr{F}(h_r^n), \tag{1.12}$$

where $\Delta\rho = 1.64$ $[g/cm^3]$ is the difference in density between the crust and water.

Leaving only the first term in the Taylor expansion above, and introducing *gravity gradient admittance* $Z_{\Gamma_{zz}}(\bar{f}) \equiv 4\pi^2 \bar{f} G \Delta\rho \exp^{-2\pi \bar{f} z_0}$, the Fourier image of the VGG can be approximated by:

$$\mathscr{F}(\Gamma_{zz}) \approx Z_{\Gamma_{zz}}(\bar{f}) \mathscr{F}(h_r^n) \tag{1.13}$$

One can then use (1.13) to obtain an expression for the topography h_r^n:

$$\mathscr{F}^{-1}(h_r^n) \approx Z_{\Gamma_{zz}}^{-1} \mathscr{F}(\Gamma_{zz}) \tag{1.14}$$

Estimation of the topography using (1.14) is what is commonly referred to as the solution of the *inverse problem*. Unfortunately, the solution of the inverse problem has a serious drawback. Since it neglects the non-linear terms, the resulting linear approximation is only valid in relatively flat areas of the ocean bottom, as indicated by Yang [1] and demonstrated in Figure 1.3

Another important effect in the linear approximation comes from the gravity gradient admittance, $Z_{\Gamma_{zz}}(\bar{f})$. As can be seen on Figure 1.4, high and low wavelengths are suppressed by the exponential factor in the gravity gradient admittance function. Thus, VGG is insensitive to those suppressed frequencies.

Thus, it is important to find another approach to the problem. The approach that was developed by Yang el al in 2018 [1] is based on solving the so-called *forward problem*, avoiding the use of Parker's series [6]. The next section describes the method used by Yang et al in some detail

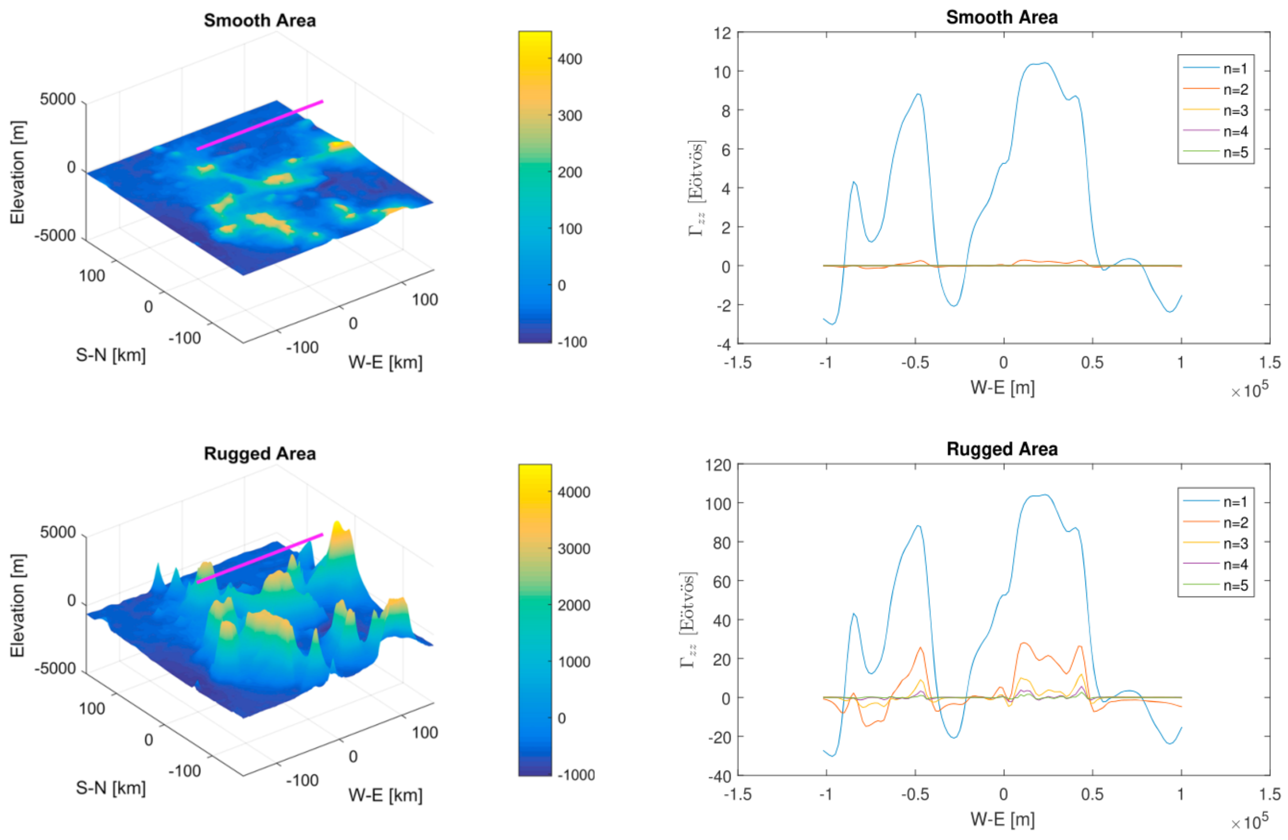


Figure 1.3: Effect of non-linear terms. Reprinted from: J. Yang, C. Jekeli, and L. Liu. "Seafloor topography estimation from gravity gradients using simulated annealing." Journal of Geophysical Research: Solid Earth 123, no. 8 (2018): 6958-6975.

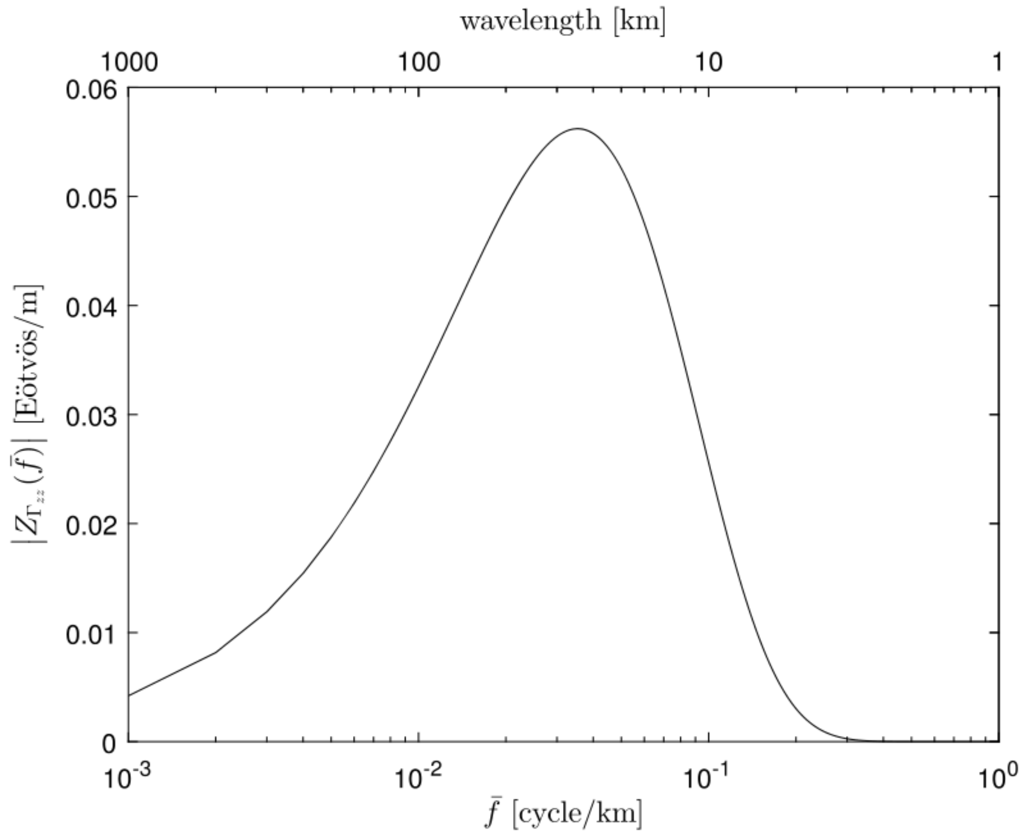


Figure 1.4: Gravity gradient admittance. Reprinted from: J. Yang, C. Jekeli, and L. Liu. "Seafloor topography estimation from gravity gradients using simulated annealing." Journal of Geophysical Research: Solid Earth 123, no. 8 (2018): 6958-6975.

since this work builds upon their work.

1.3 Forward problem

Yang managed to go beyond the linear approximation by using a global optimization technique known as Simulated Annealing (SA). The essence of his idea is to explore the parameter space of the topography and find the set of parameters that minimize the difference between the calculated VGG and the measured VGG.

The parametrization they chose lead to a very large search space. The reason is that they represent the topography as a combination of rectangular prisms and calculate VGG for each prism. This leads to the huge search space with dimensionality on the order of 10^4.

CHAPTER II

TOPOGRAPHY ESTIMATION USING RADIAL BASIS FUNCTIONS

The main idea of the forward modeling method is to calculate the VGG of candidate topographies and compare them with a VGG coming from direct measurements. Then the topography that produces the VGG minimizing the root mean square difference from the 'true' VGG is selected.

Thus, further developing the idea of Yang [1], who kindly provided the patch of an ellipsoid, as well as the topographic and VGG data, the forward problem has been approached in the following way:

1. Topography is generated as a sum of linear combination of Radial Basis Functions (RBF) and a patch of an ellipsoid representing the geoid. RBFs are simply 2-dimensional Gaussians. Each RBF has four parameters - standard deviation σ, height h and coordinates of its center (mean), X_{mean} and Y_{mean}. This greatly reduces the dimensionality of the search space: from 10^4 in Yang's work [1] to 1500 in our case (see chapters 5 and 6 of this thesis).

2. VGG is calculated for each candidate topography. The calculation is done using Parker's series to the first order (linear approximation), equation 1.12.

3. The resulting VGG is compared with VGG from data. The comparison is done using the RMS difference:

$$RMSE = \mu[(VGG_{data} - VGG_{design})^2], \tag{2.1}$$

where μ denotes the mean.

4. The topography that produces VGG closest to the VGG from data, i.e. minimizing the RMS difference, is selected as the best estimate of the true topography.

There are two main differences between our work and previous work by Yang. First, we use a different parametrization of the topography. In Yang's work [1], the topography is parametrized as a set of rectangular prisms. As a result, the parameter search space is very large, with dimensionality on the order of 10^4. Our method reduces the dimensionality to 3 $\times$ Number of RBFs. This is because there are only 3 parameters that are minimized over computationally, since minimization over height can be done analytically.

Analytic computation of the heights is possible because the model depends on the heights linearly. More specifically, calculation of the VGG involves Fourier transformations and Taylor expansion, as brief examination of equation 1.12 shows (assuming only the first term is kept). Now, it is well known that Fourier transformation is a linear operator. Same is true for the series expansion, again assuming the linear order approximation. Therefore, the problem of finding the optimal heights can be formally cast as follows.

The fitness function is a linear combination of design matrices $B_{i,j}$, with coefficients given by heights h_j, which can be represented by a row vector $\bar{h} = (h_0, h_1, ..., h_{M-1})$. Design matrix is a first term of the Taylor expansion of Fourier transforms of a seafloor patch (seafloor patch consists of RBFs plus the ellipsoid patch), as per equation 1.12. Therefore, one obtains (see [7], Appendix C.2, p.109):

$$f(\vec{r}) = \sum_{j=0}^{M-1} h_j B_{j,i}(\vec{r}, \bar{b}) = \bar{h}\mathbf{B}, \tag{2.2}$$

where M is the number of RBFs used in generating the design matrix of the topography.
Hence, finding the least squares difference amounts to:

$$L_s = ||\bar{y} - \bar{h}\mathbf{B}||^2 = [||\bar{y}||^2 - 2\bar{y}\mathbf{B}^T\bar{h}^T + \bar{h}\mathbf{B}\mathbf{B}^T\bar{h}^T] \tag{2.3}$$

Now, denoting the optimal heights by $\hat{h}$, the extremum of that function can be found using the standard method from calculus:

$$\left.\frac{\partial L_s}{\partial h_i}\right|_{\hat{h}} = 0 \implies \hat{h} = \bar{y}\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1} \tag{2.4}$$

The quantity $\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ is known as the Moore-Penrose inverse of a matrix. Therefore, denoting the Moore-Penrose inverse as pinv (following the notation used in MATLAB), one obtains the following simple formula for the minimizing array of heights:

$$\hat{h} = \bar{y} \cdot pinv(\mathbf{B}) \tag{2.5}$$

Unfortunately, if applied as given, equation (2.5) gives an incorrect result: the analytic estimate of the heights picks up an overall shift along the z axis that seems to vary unpredictably. Following the computation step by step has shown that the shift occurs due to the inversion problem being severely ill-conditioned, with MATLAB reporting a condition number on the order of 10^{18}. To correct for the shift, we subtract the average of the topography values from the estimate and then subtract 4500 m more to shift it to the proper location on the z axis:

$$T = ET - \mu[ET] - \mu[d], \tag{2.6}$$

where:

T - final topography estimate,

ET - estimated topography (before the correction for the shift),

$\mu[ET]$ - mean of the estimated topography,

$\mu[d]$ - mean sea depth, $\mu[d]$ = 4500 m.

This is only a preliminary solution. The next stage of the project is a search for a basis that does not lead to an ill-conditioned problem. Presently, the method described above seems to be acceptable on the grounds that it is well-defined and solves the problem at hand sufficiently well as will be demonstrated in chapter 4.

Next, the remaining three parameters should be minimized over numerically. The problem of finding the minimum value of a function over a set of parameters is called an *optimization* problem, and the function is referred to as the *fitness function*. Thus, finding the set of parameters of a topography that minimizes the RMS difference between the computed and observed VGG is an optimization problem. This leads to the second difference of our work from previous attempts at solving the forward problem - the optimization method. Yang, for example, used Simulated Annealing (SA) as an optimization method. But the problem at hand is a challenging multidimensional non-convex optimization problem. As a result, the use of SA leads to several-hour long computation on a super computer. Therefore, we decided to utilize a stochastic optimization algorithm, that doesn't necessarily explore the whole of the search space and is well suited for continuous variable search spaces: *particle swarm optimization (PSO)*.

Next chapter provides an overview of various types of optimization problems and algorithms developed to tackle those problems with particular emphasis on PSO.

CHAPTER III

PARTICLE SWARM OPTIMIZATION

In this chapter an overview of the Particle Swarm Optimization method is provided. The discussion will be based on a book written by my advisor, Dr. Soumya D. Mohanty [7].

3.1 Optimization problems

In general, optimization problems require one to locate a maximum or a minimum of a given function. They form a large subset of problems encountered in many areas of physics. For instance, problems of finding minima of energy, maxima of entropy, and minimizing "action" all belong to this subset.

The function to be minimized is typically referred to as a *fitness function* f($\bar{x}$), $\bar{x} \in \mathbb{D} \subseteq \mathbb{R}^D$, where D is the dimensionality of the *search space* $\mathbb{D}$. Thus, optimizing a function is formally defined as the search for an *optimizer*, i.e. finding $\bar{x}^*$ such that *f($\bar{x}^*$)* has a minimal (maximal) value. Among great many different approaches to solving optimization problems, *stochastic algorithms* play an important role. The general idea of stochastic algorithms is to avoid searching the whole of the search space, since as the dimensionality increases, the size of the search space grows exponentially and the program of testing every point in it quickly becomes unfeasible computationally.

Stochastic algorithms can generally be described as follows. First, consider the vector $\bar{x}$[k] that represents the set of parameters (point in the search space) at step k. Then, the algorithm is:

1. *Initialization* step: k is set to zero and a random point is chosen in the search space, i.e. $\bar{x} \in \mathbb{D} \subseteq \mathbb{R}^D$ is selected from some specified random distribution.

2. *Randomization* step: a *velocity* vector is drawn from some random distribution (the distribution can change at every iteration), i.e. $\bar{V}$ is a vector random variable described by a joint

probability density function $P_{\bar{V}}$ ($\bar{x}$, k).

3. *Position update* step: change the old position using the random velocity vector above according to some rule, i.e. $\bar{x}$[k+1] = f($\bar{x}$[k],$\bar{v}$[k])

4. *Update pdf* step: $P_{\bar{V}}$ ($\bar{x}$, k) is changed to $P_{\bar{V}}$ ($\bar{x}$, k + 1)

5. *Update* k to k+1

6. *Repeat* until some *termination* condition is satisfied.

Now, within the family of stochastic algorithms, *evolutionary* algorithms plays an increasingly important role. As the name suggests,they are generally inspired by biological evolution and have a distinctive feature of employing a *population of agents* that do the searching and can 'communicate' with each other. Evolutionary algorithms are typically divided into *genetic* algorithms and *swarm intelligence* algorithms. The first category involves algorithms that arouse out of direct attempts at simulating biological evolution on computers, while the second is chiefly represented by *Particle Swarm Optimization* (PSO).

3.2 Particle Swarm Optimization

PSO is a powerful stochastic optimization algorithm that is especially well suited to the continuous variable multidimensional optimization problems. Despite following the general procedure described above, PSO has several distinctive features which are the result of it being modelled after the behaviour of swarms of insects or flocks of birds.

Like all evolutionary algorithms, PSO consists of a population of agents ('a swarm of particles') that move around the search space and measure the value of the fitness function at each location of a particle. At every iteration each particle gets a 'kick', represented by a 'velocity' vector. Different versions of PSO use different rules for updating the velocity vector at each iteration. These rules are commonly called the *dynamical equations* of the PSO. We will only consider the version of the PSO that was used in this work and also happens to be the most efficient PSO variant for most problems - the *local best* PSO.

Any PSO algorithm has two major parts: *kinematics* and *dynamics*. First the kinematics of the local best PSO is to be presented.

3.2.1 Kinematics of the Local Best PSO

Let us start by establishing notation used in PSO. It is presented in Table 1 below, reproduced from [7], p. 47.

Symbol	Description
N_{part}	The number of particles (agents) in the swarm
$\bar{x}^{(i)}$[k]	Position of the i^{th} particle at the k^{th} iteration
$\bar{v}^{(i)}$[k]	Velocity of the i^{th} particle at the k^{th} iteration
$\bar{p}^{(i)}$[k]	Best location found by the i^{th} particle (particle best or pbest)
$\bar{l}^{(i)}$[k]	Best location found in the neighborhood of the i^{th} particle (local best or lbest)
$\bar{g}$[k]	Best location found by the swarm (global best or gbest)

Table 1. Notation used in PSO

The kinematics of the local best variant of PSO (lbest) is represented by quantities used to describe the position of the particle in the search space and the points in it that are used when updating the position. The position is represented by a vector $\bar{x}^{(i)}$[k], the change in position is described by the velocity vector $\bar{v}^{(i)}$[k] (in strict physics terminology that would be *displacement*). At every iteration the velocity vector is updated such that each particle feels 'attraction' to the pbest point and to the lbest point in the search space with some random weights. The details of these update rules are given in the next secion on the dynamics of PSO.

Before trying to understand the dynamics, however, one has to clearly understand the idea of a *particle neighborhood.* The neighborhood of the i^{th} particle is defined as a subset $\mathbb{N}^{(i)}$ of a set of all indices of particles $\mathbb{I}^{(i)}$, including i^{th} index itself. That is to say:

$\mathbb{N}^{(i)} = \mathbb{I}^{(i)} \cup \{i\}$

$\mathbb{I}^{(i)} = \{\ 1,2,..., N_{part}\} \setminus \{i\}$,

where $\mathbb{A} \setminus \mathbb{B}$ denotes *absolute complement* of $\mathbb{B}$ in $\mathbb{A}$ (sometimes called their *difference*), i.e. all elements in $\mathbb{A}$ that are not in $\mathbb{B}$. For instance, if $\mathbb{A} = \{1,2,3\}$ and $\mathbb{B} = \{3,4,5\}$, then $\mathbb{A} \setminus \mathbb{B} = \{1,2\}$.

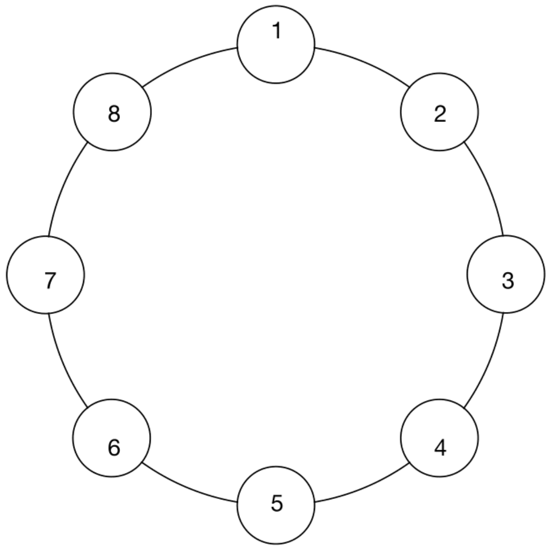


Figure 3.1: Ring topology, adapted from: Soumya D. Mohanty, Swarm Intelligence Methods for Statistical Regression, Chapman and Hall/CRC, 2018, p. 58

Now, using the concept of the neighborhood, one can define the local-best particle as the particle in the neighborhood of the i^{th} particle at the k^{th} iteration that has the minimum value of the fitness function. By providing a particular $\mathbb{N}$ one is said to have chosen a *topology* for the problem.

In this work the so-called *ring* topology was utilized. It is given by:

$$\mathbb{N}^{(i)} = \{r(i-k); \quad -m \leq k \leq m, \quad m \geq 1$$

$$r(j) = j \quad \textit{if} \quad 1 \leq j \leq N_{part}$$

$$r(j) = j + N_{part} \quad \textit{if} \quad j < 1$$

$$r(j) = j - N_{part} \quad \textit{if} \quad j > N_{part}$$

In this topology the neighborhood size is $2m+1$. In order to understand the ring topology it is best to look at the figure 3.1. In this figure, if $m = 1$ then the neighborhood size is 3. Therefore, the neighborhood of, say, particle 1 includes particles 8,1 and 2.

The particle best value is the best value found by the particle in question at iteration k:

$f(\bar{p}^{(i)})$ [k] = $\min\limits_{j \leq k} f(\bar{x}^{(i)})[j]$

The global best value is the best value found by the swarm at iteration k:

$f(\bar{g})$ [k] = $\min\limits_{j} f(\bar{p}^{(j)})[k]$

Now, having defined the terms, we can proceed to consider the dynamics of the lbest PSO.

3.2.2 Dynamics of the Local Best PSO

Dynamics of the lbest PSO is given by a set of rules governing the updates of the velocity and positon. The *velocity update* rule is given by:

$$\bar{v}^{(i)}[k+1] = \omega\bar{v}^{(i)} + \mathbf{R}_1 c_1 \bar{F}_c^{(i)}[k] + \mathbf{R_2} c_2 \bar{F}_s^{(i)}[k],$$

where $\bar{F}_c^{(i)}[k]$ is the *cognitive* term, $\bar{F}_s^{(i)}[k]$ is the *social* term, c_1 and c_2 are *acceleration constants*, ω is the *inertia weight* and $\mathbf{R_1}$ and $\mathbf{R_2}$ are random diagonal matrices. Definitions of these terms are given below.

Incidentally, the velocity is *clamped*, i.e. there is a maximum value for the magnitude of the components of velocity, denoted by v_{max}.

Next, the *position update* is done according to:

$$\bar{x}^{(i)}[k+1] = \bar{x}^{(i)}[k] + \bar{v}^{(i)}[k+1]$$

Now, the definitions of the terms described above are to be provided.

The *cognitive* term is a vector pointing towards the pbest location for each particle and acts as a 'force' pulling the particle towards that location. It is given by: $\bar{F}_c^{(i)}[k] = \bar{p}^{(i)}[k] - \bar{x}^{(i)}[k]$

The *social* term, in contrast, pulls the particle towards the local best location in the particle's neighborhood. It can be found using: $\bar{F}_c^{(i)}[k] = \bar{l}^{(i)}[k] - \bar{x}^{(i)}[k]$

The *inertia weight*, ω, is a positive real number that, if large, tends to keep the velocity in the same direction, hence the name.

Constants c_1 and c_2 are referred to as *acceleration* constants and affect the influence of the social and cognitive terms on the velocity.

Finally, the $\mathbf{R_i}$, $i = 1,2$, is a diagonal matrix, with each diagonal element being an independent random variable with uniform distribution in the range from 0 to 1.

3.2.3 Boundary conditions

In the context of PSO, *boundary conditions (BC)* are the rules used when the particle reaches the boundary of the search space. There are several boundary conditions that are commonly used in PSO.

Let them fly is the boundary condition that was used in this work and is the simplest and yet seemingly the most effective BC available. The rule is simply to assign a fitness value of ∞ to the particle once it leaves the search space (or $-\infty$ in case the problem is that of maximization rather than minimization). As it turns out, the particle will eventually return back into the search space. This happens because once it's fitness is set to ∞, it can no longer affect the swarm's gbest or lbest values nor it's own pbest value. Therefore, it is gradually pulled back by the attraction of the lbest and pbest locations.

CHAPTER IV

RESULTS

4.1 Results obtained with user-generated random topographies

User generated data is based on a script that produces randomly generated topographies from a given number of RBFs. Figures 1 - 3 show the topographies generated along with the inferred ones using the VGG codes (described in the Appendix A).

The results of the PSO run with 200 iterations and 30 RBFs used for the generation of the topography is shown in figures 4.3 and 4.4. The RMSE error normalized by the range is 14%.

The results of the PSO run with 200 iterations and 500 RBFs used for the generation of the topography is shown in figures 4.3 and 4.4. The RMSE error normalized by the mean is 15.8%.

4.2 Results obtained with real data

In this chapter I present the first results obtained with real data (from Dr. Junjun Yang, email correspondence). The data source, quoting from [1], is provided in the section on 'Acknowledgements' of that paper and reads: "The vertical gravity gradient data and the SIO global topography models used in this paper are available at ftp://topex.ucsd.edu/pub/. The ship sounding depths can be downloaded from www.ngdc.noaa.gov/maps/bathymetry."

The topography data is shown in figures 4.5 and 4.6 (both figures present the same topography from different view points).

The VGG data is shown in figures 4.7 and 4.8.

Figures 4.9 and 4.10 show the topography and the VGG it produces together for comparison.

Figures 4.11 and 4.12 show the topography estimation results using PSO with 500 RBFs and 200 iterations. RMSE normalized by the range and mean are both around 16%.

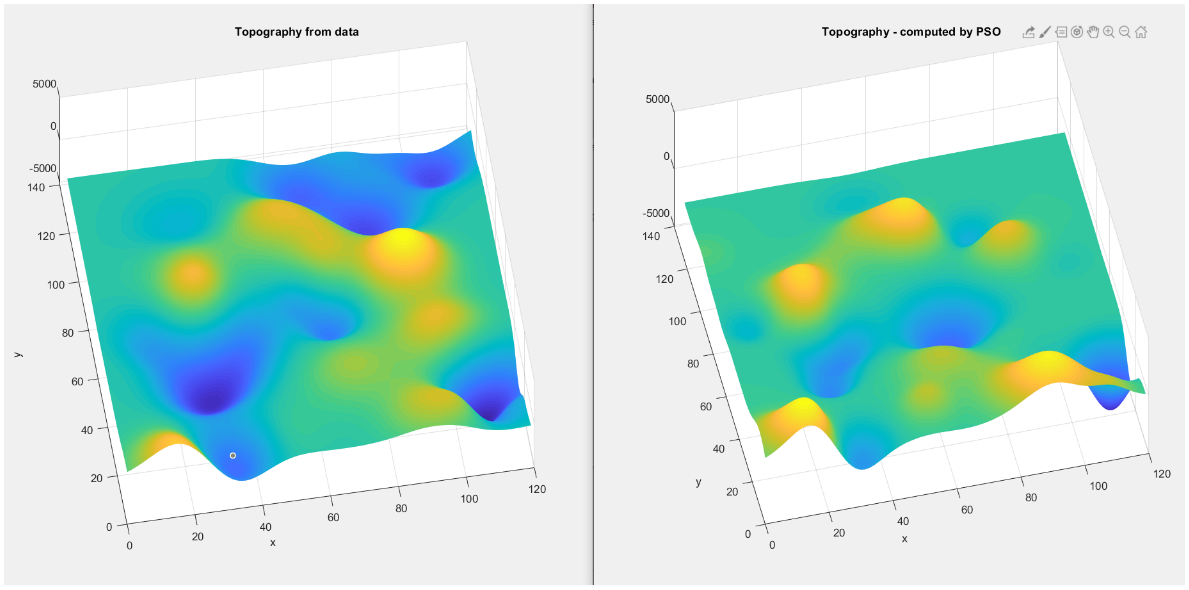


Figure 4.1: Topography with 30 RBFs (left) and its PSO estimate (right) - angled view

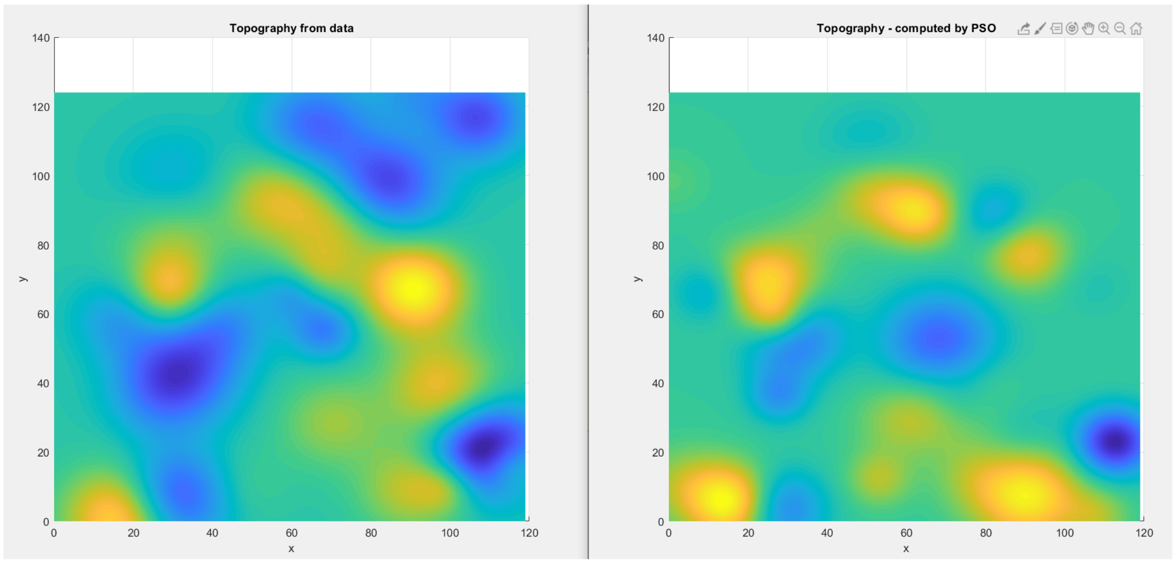


Figure 4.2: Topography with 30 RBFs (left) and its PSO estimate (right) - top view

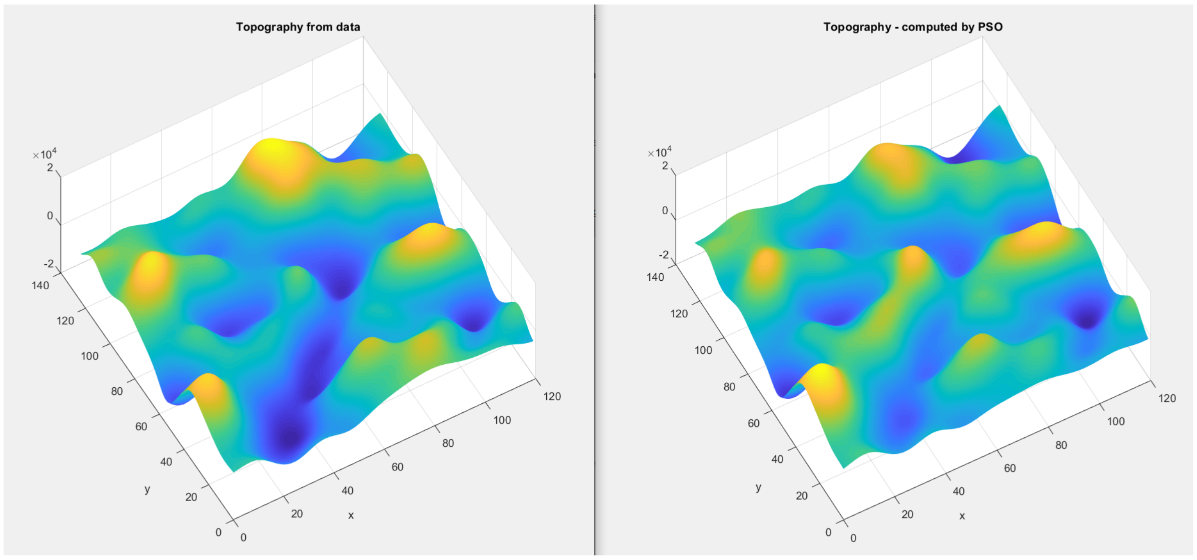


Figure 4.3: Topography with 500 RBFs (left) and its PSO estimate (right) - angled view

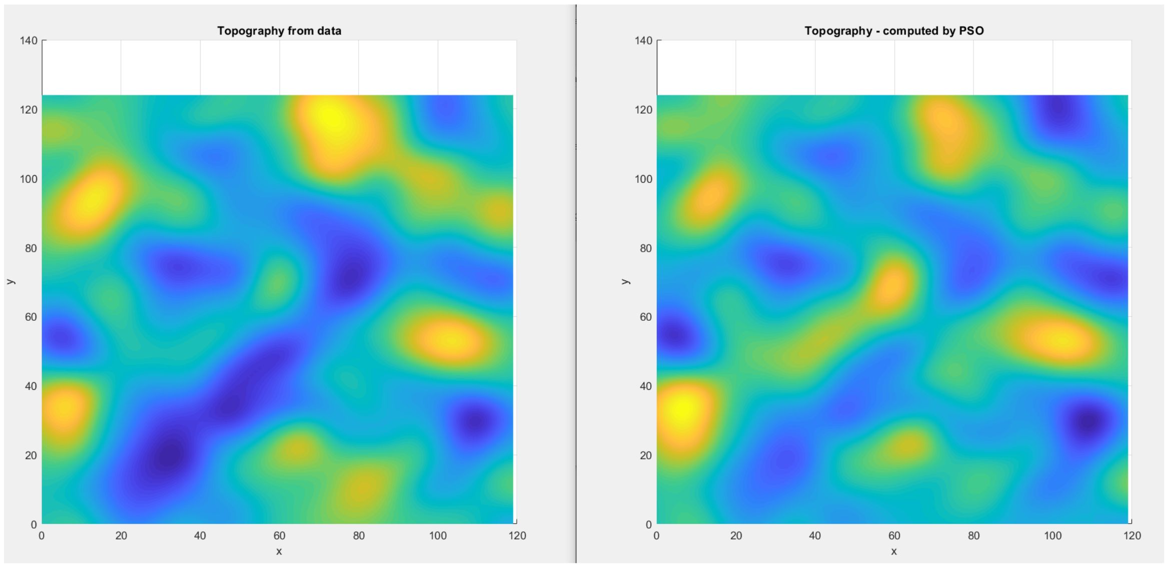


Figure 4.4: Topography with 500 RBFs (left) and its PSO estimate (right) - top view

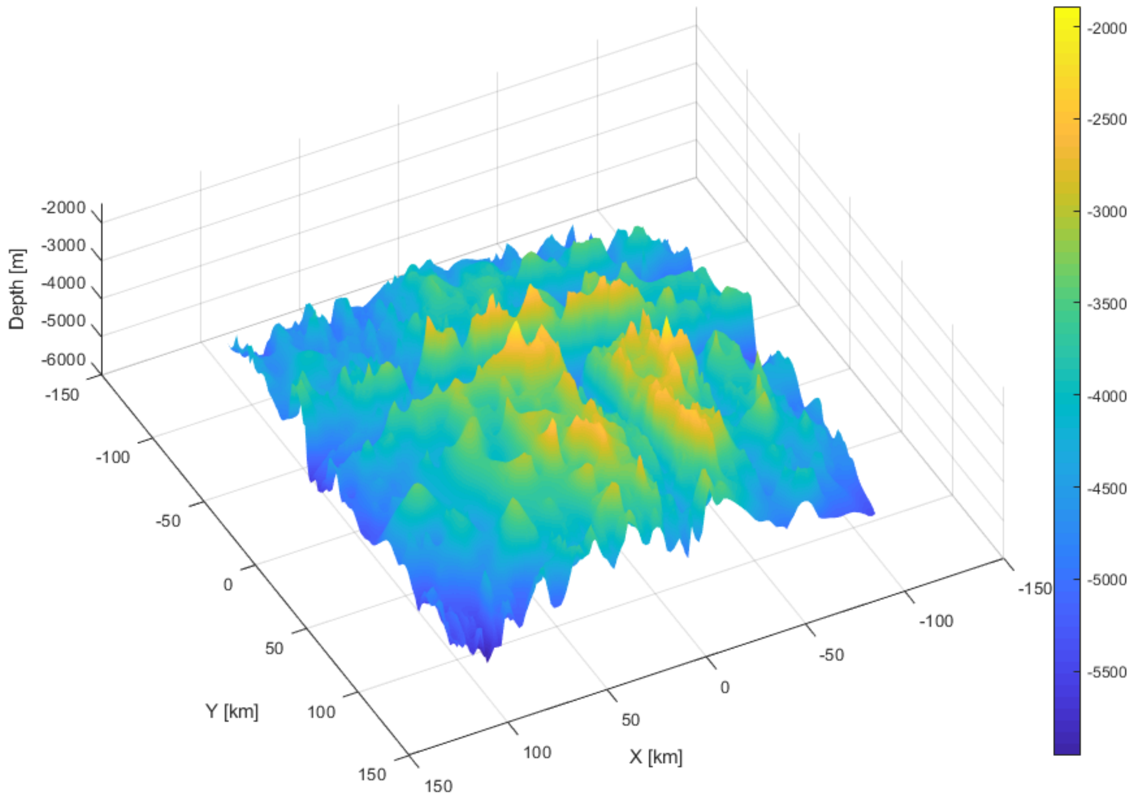


Figure 4.5: Topography data, provided by Dr. Junjun Yang

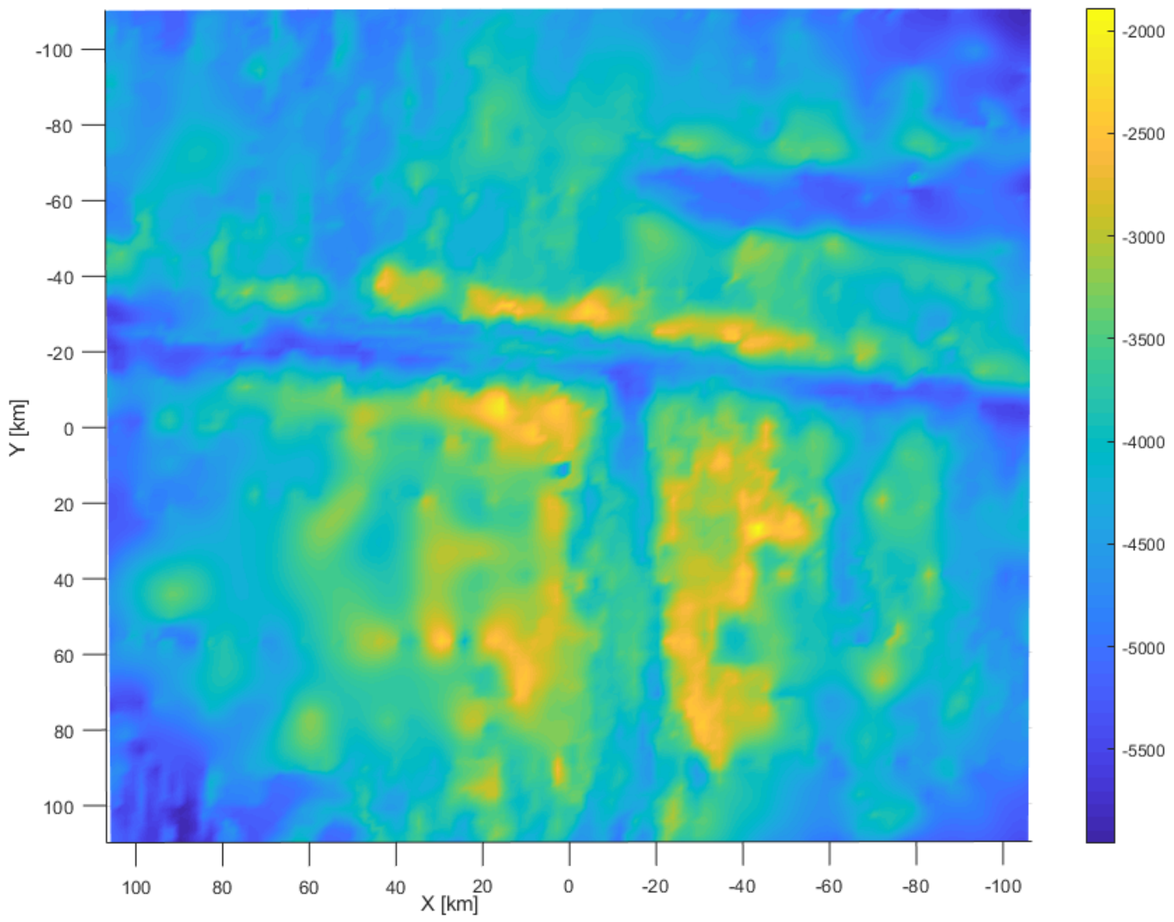


Figure 4.6: Topography data, top view. Depth on the colorbar is in meters.

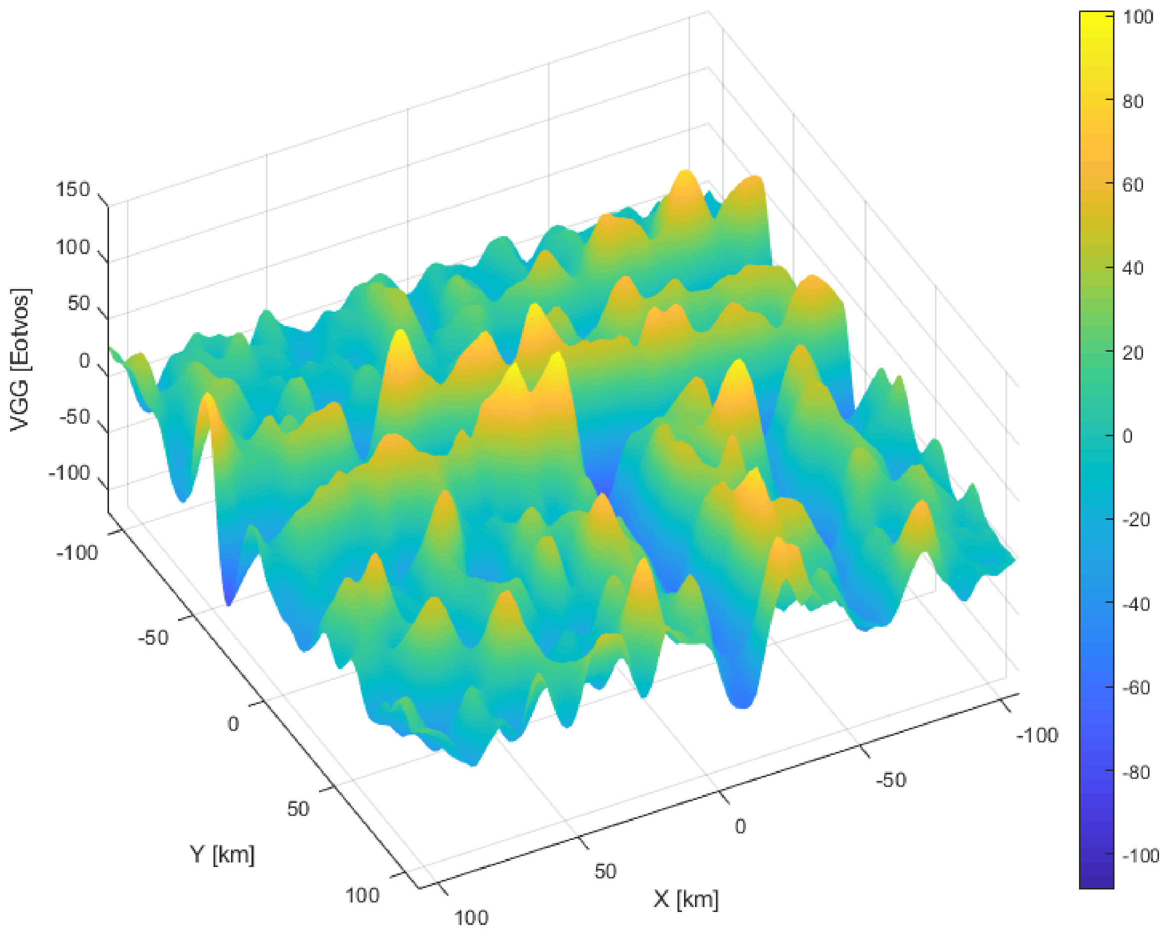


Figure 4.7: VGG data, provided by Dr. Junjun Yang

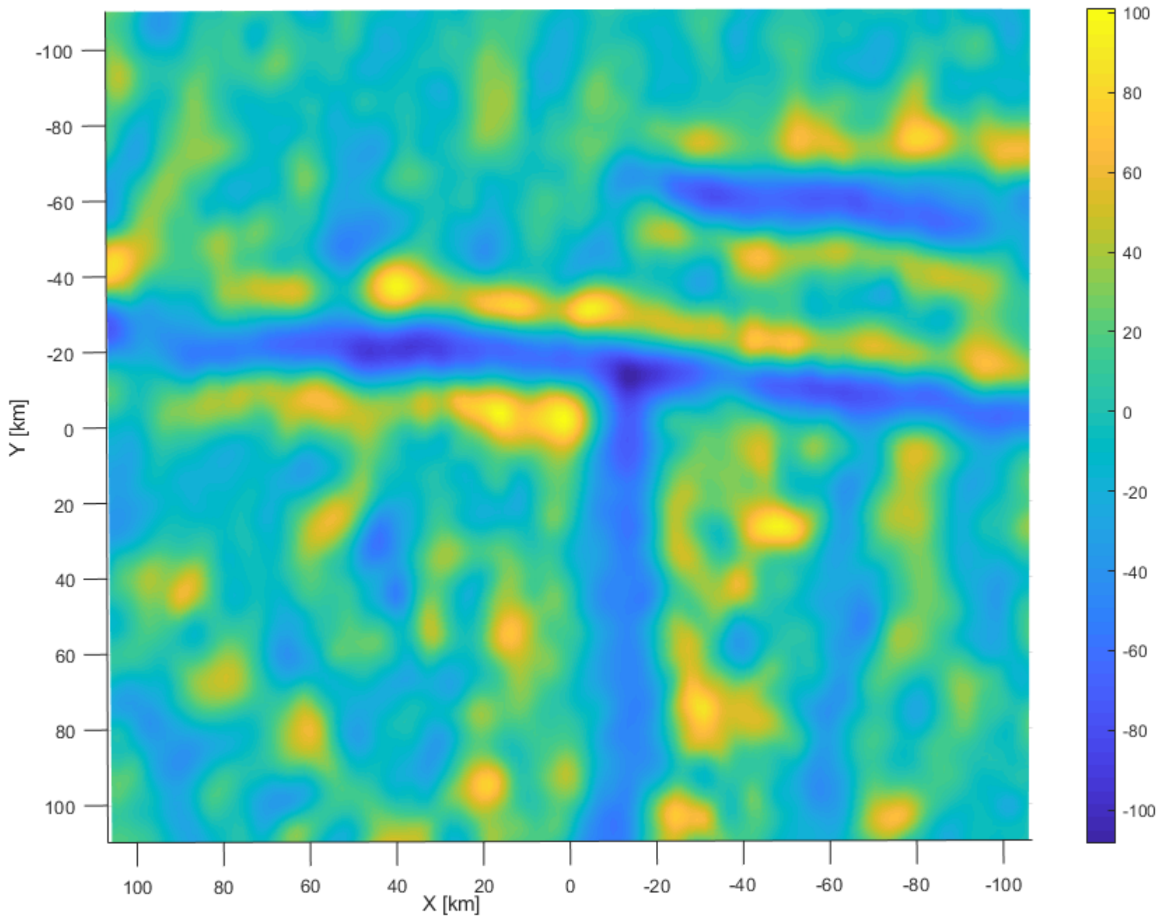
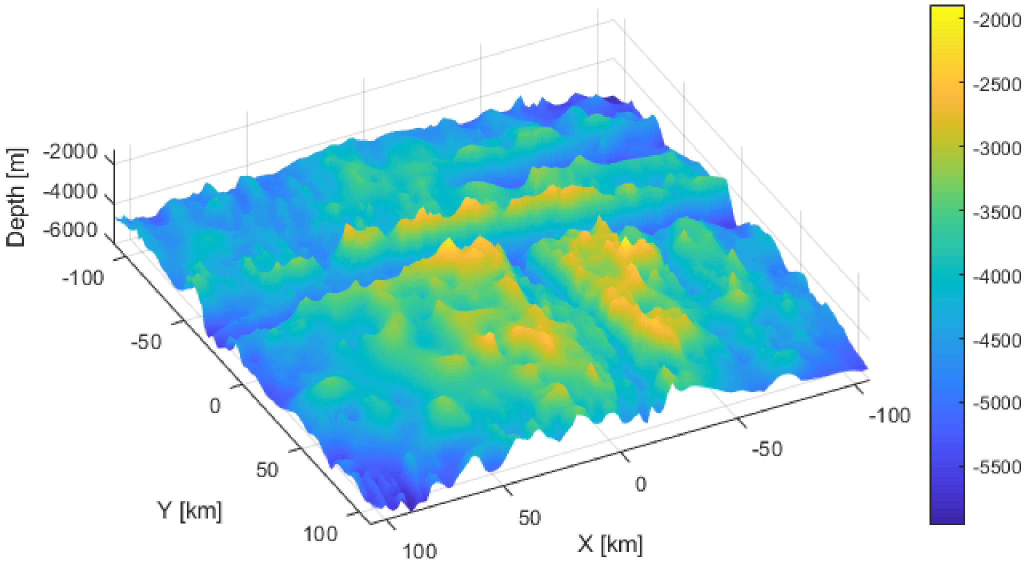


Figure 4.8: VGG data, top view. Colorbar is in Eotvos.



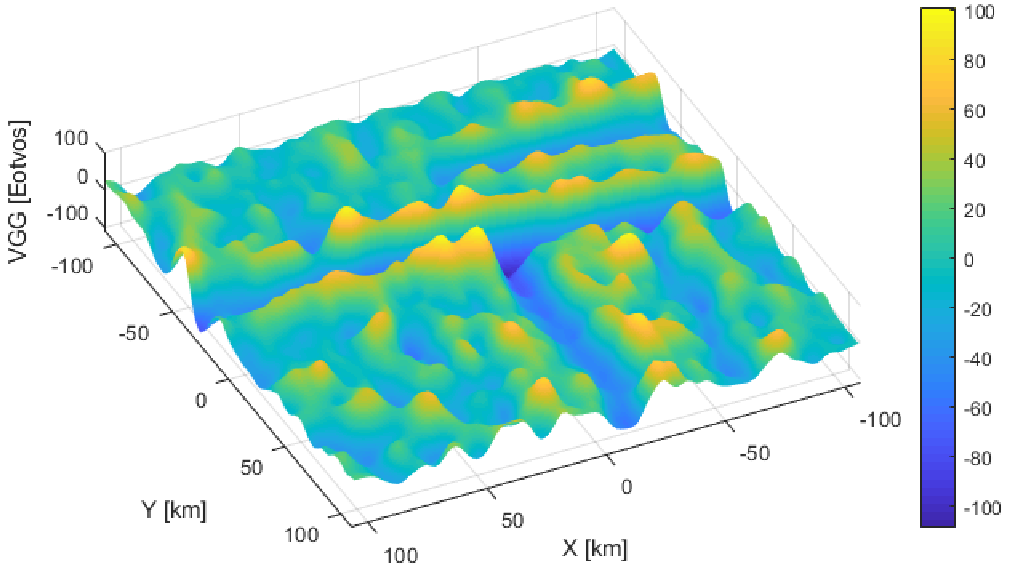


Figure 4.9: Topography (on top) and its VGG (on the bottom), view at an angle

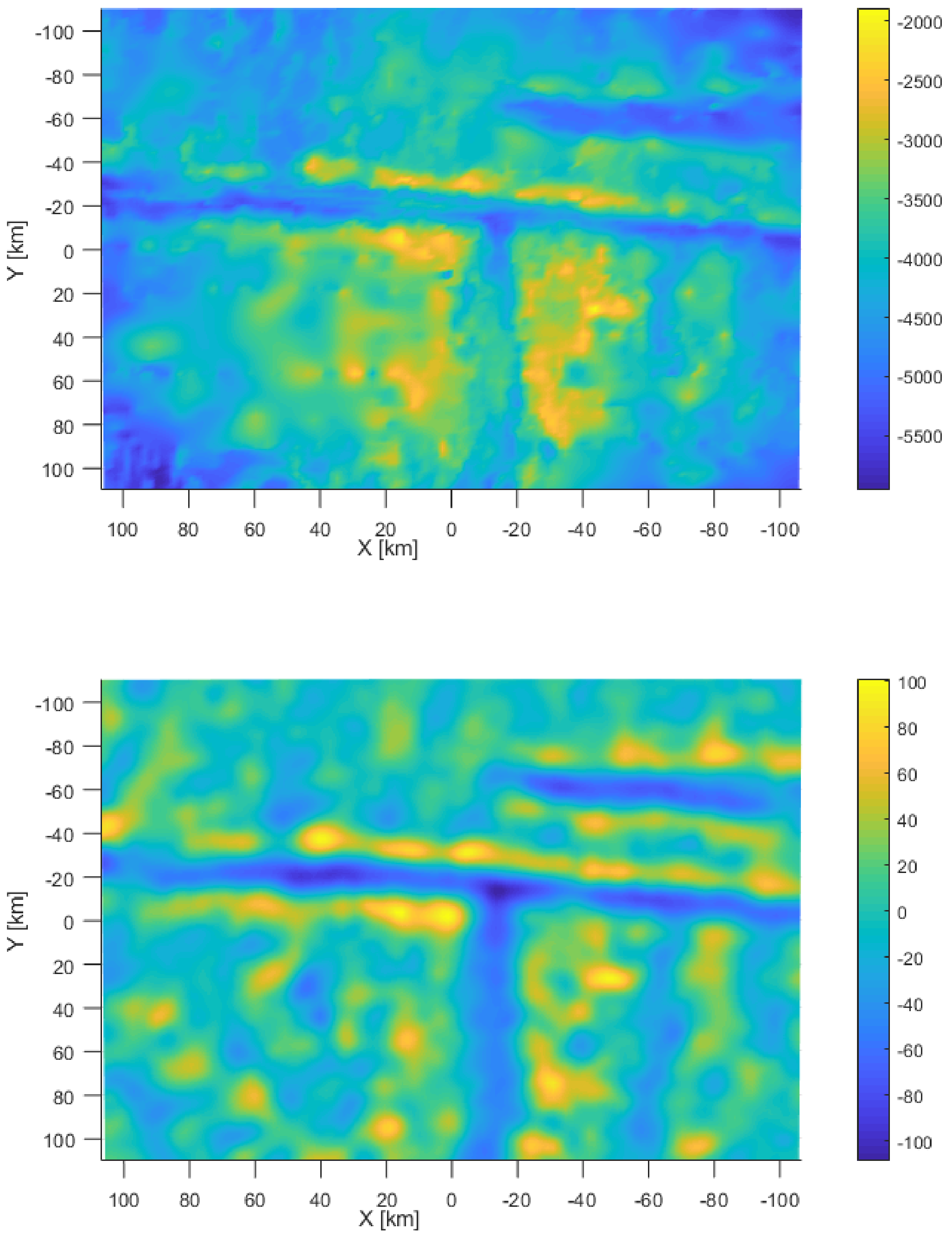


Figure 4.10: Topography (on top) and its VGG (on the bottom), top view. Colorbar is in meters (top) and in Eotvos (bottom)

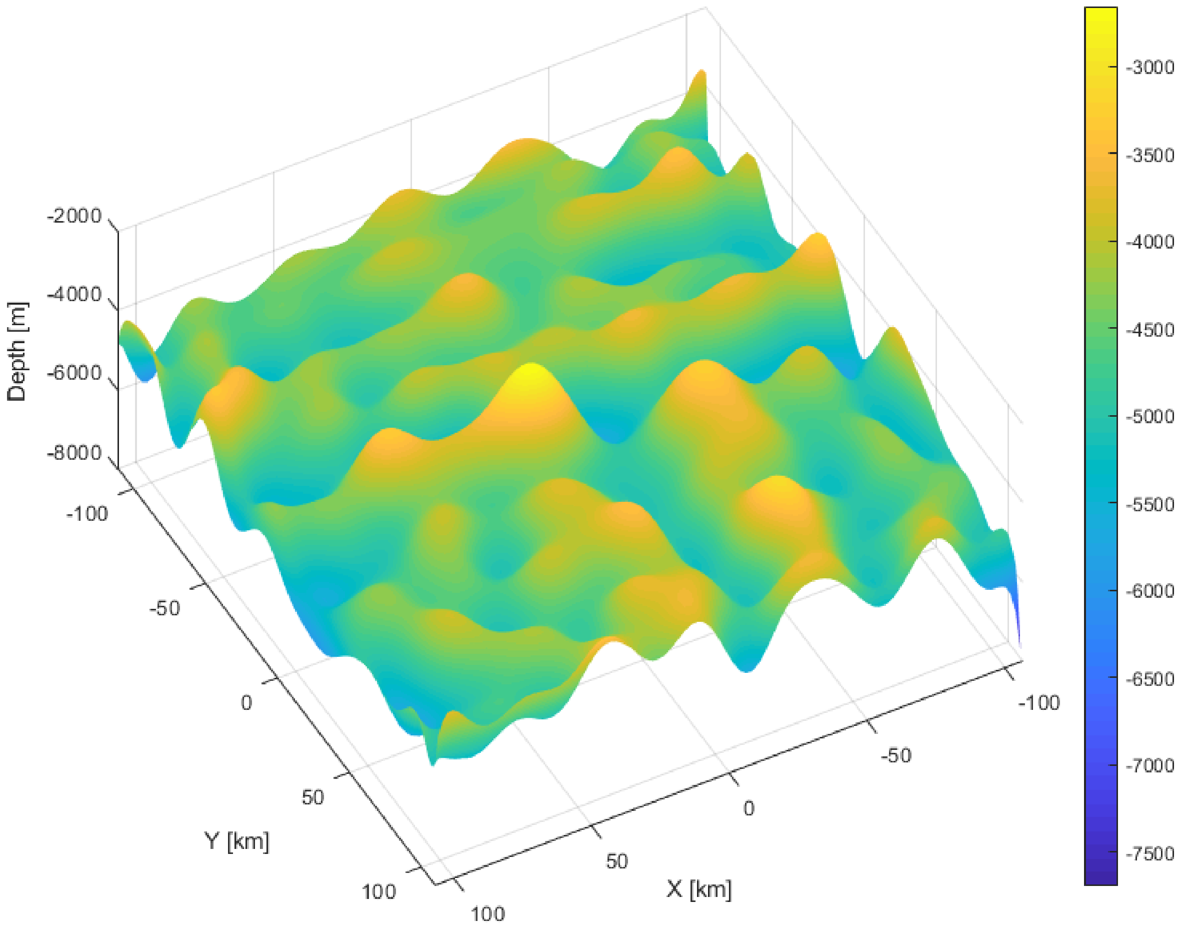


Figure 4.11: Topography estimated by PSO using 500 RBFs, view at an angle, RMSE $\approx 16\%$

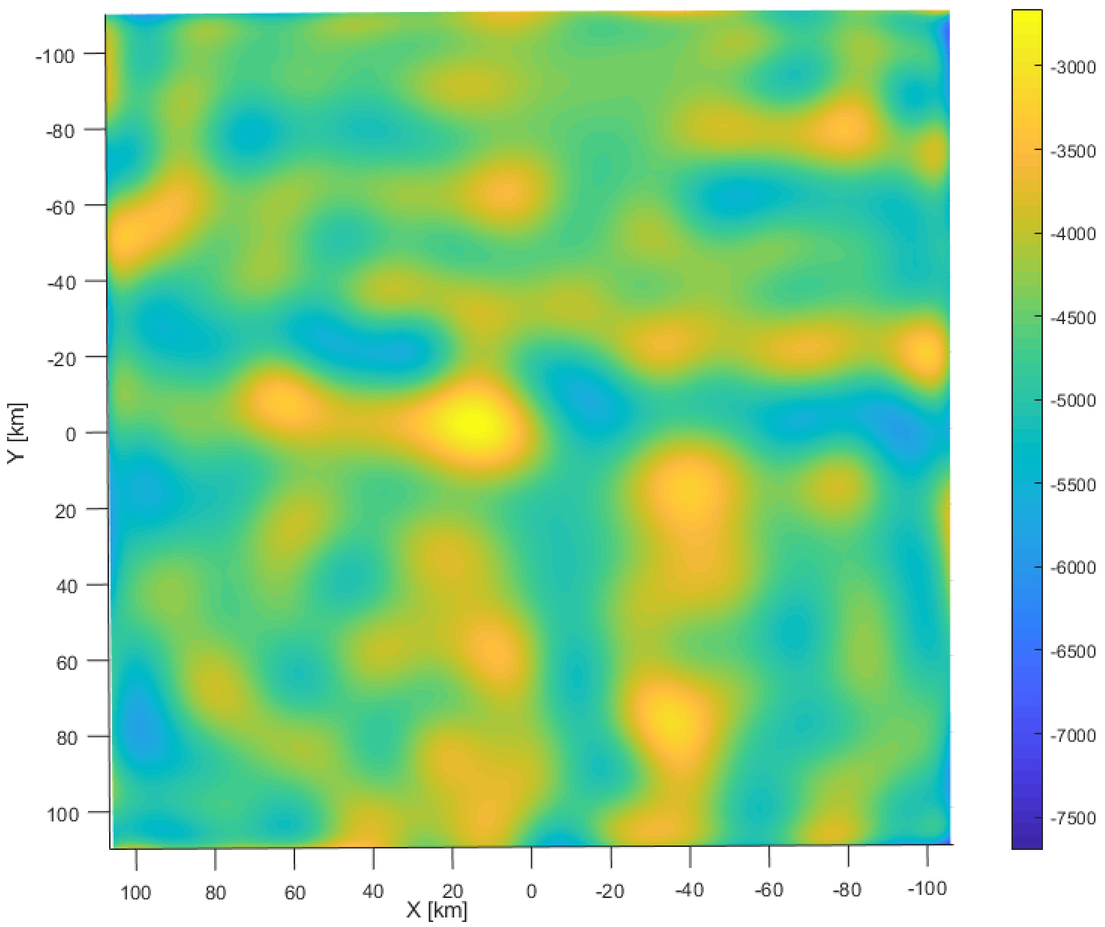


Figure 4.12: Topography estimated by PSO using 500 RBFs, top view, RMSE $\approx 16\%$. Colorbar is in meters

Figures 4.13 and 4.14 show the estimated and ’true’ topographies together for ease of comparison.

Figure 4.15 shows the PSO estimated topography with 1500 RBFs used in the design matrix and 200 PSO iterations. The RMSE has decreased to $\approx$15.8%.

Figure 4.16 shows the PSO estimated topography with 3000 RBFs and 200 PSO iterations. The RMSE has increased to $\approx$17%.

4.3 Discussion

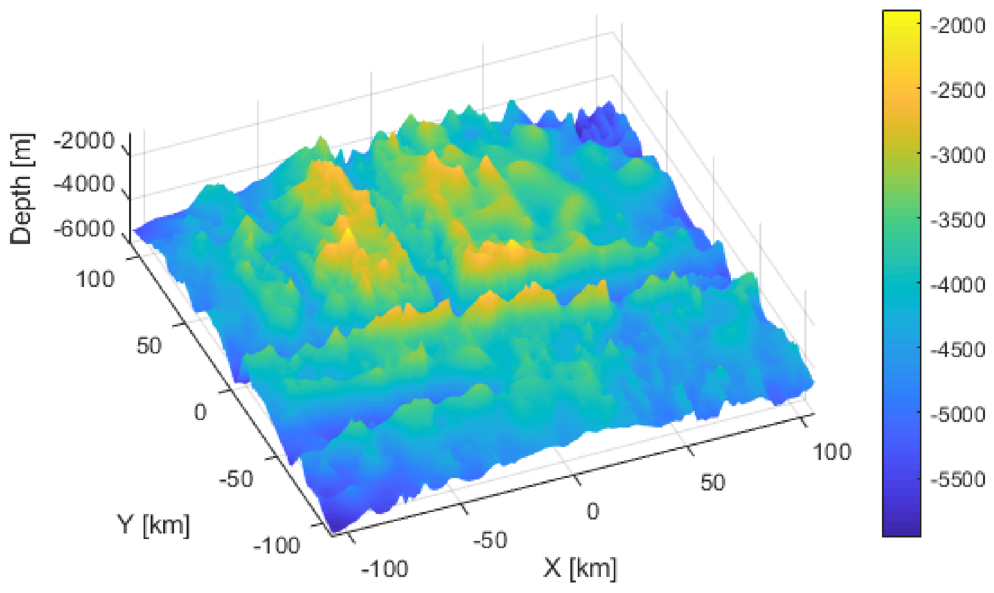
The first results appear to be very encouraging. First of all, RMS errors normalized by the mean/range are around 14% for user-generated data, and 16% ($\approx$ 650 m) for real data. Comparing that with Yang’s results, who had an RMS around 11% ($\approx$300 m) and taking into account the fact that Yang et al used super-computers and their calculation took many hours as opposed to $\approx$ 30 min long calculations on a personal laptop for 500 RBFs and around 2 hours for 3000 RBFs, we hope that future refinements of the codes will bring state of the art performance levels.

Increasing the number of RBFs used in generating the topographies does not seem to have a strong effect on performance, as figures 4.15 and 4.16 clearly show, since precision improved only slightly in the 1500 RBF case (RMSE 15.8%) and even decreased for 3000 RBFs (RMSE 17%).

Future work will be focused on improving the current method. There are several potential improvements to the current approach. First of all, one should try to use an exact formula for the computation of the VGG from topography. The formula is straightforward in case of rectangular prisms used by Yang et al [1], but it is still under development for the case of RBFs.

Second, one has to try to limit the standard deviation range because high and low frequencies are suppressed in the Parker series, as can be seen in 1.12 and discussed in Chapter 1. This should lead to the reduction of the size of the search space and thus make the computation considerably faster.

Finally, one can try to estimate the positions of RBFs from the solution of the inverse problem and hence let PSO start in the part of the search space that is likely to contain the minimum.



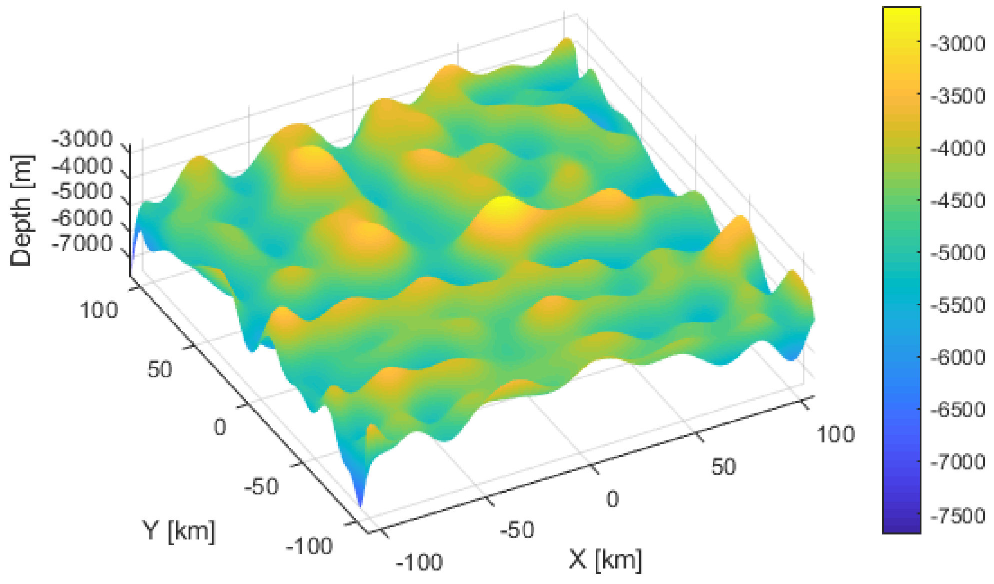
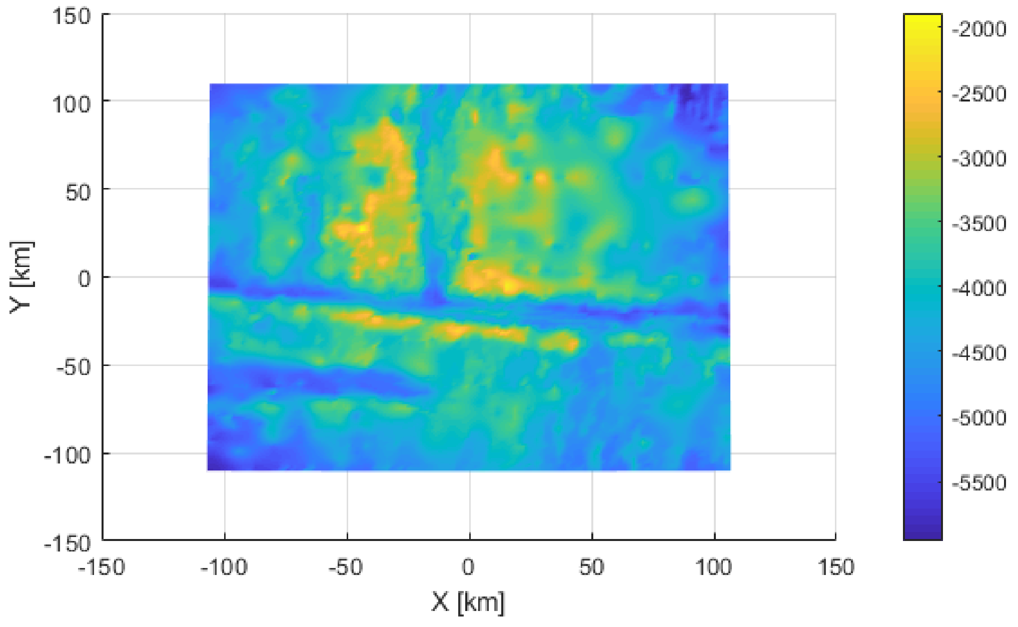


Figure 4.13: Estimated (bottom) and true topography together, 500 RBFs, RMSE $\approx 16\%$



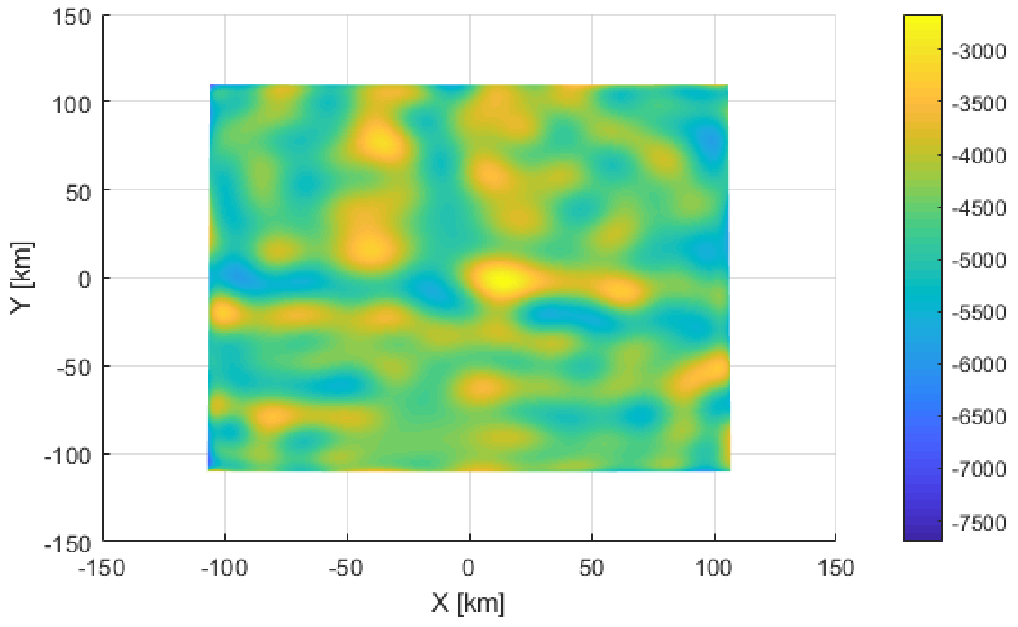


Figure 4.14: Estimated (bottom) and true topography together, top view, 500 RBFs, RMSE $\approx$ 16%. Colorbars are in meters.

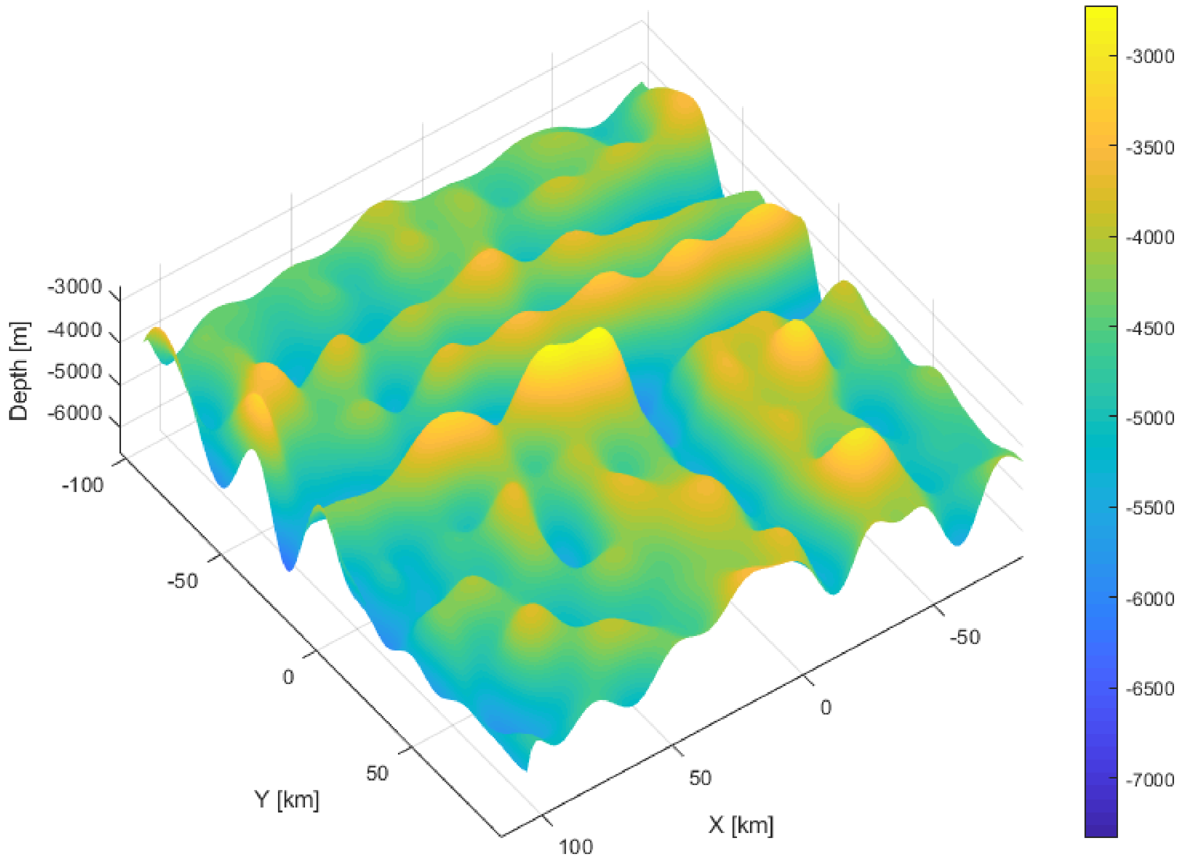


Figure 4.15: PSO estimated topography, 1500 RBFs, RMSE ≈ 15.8%

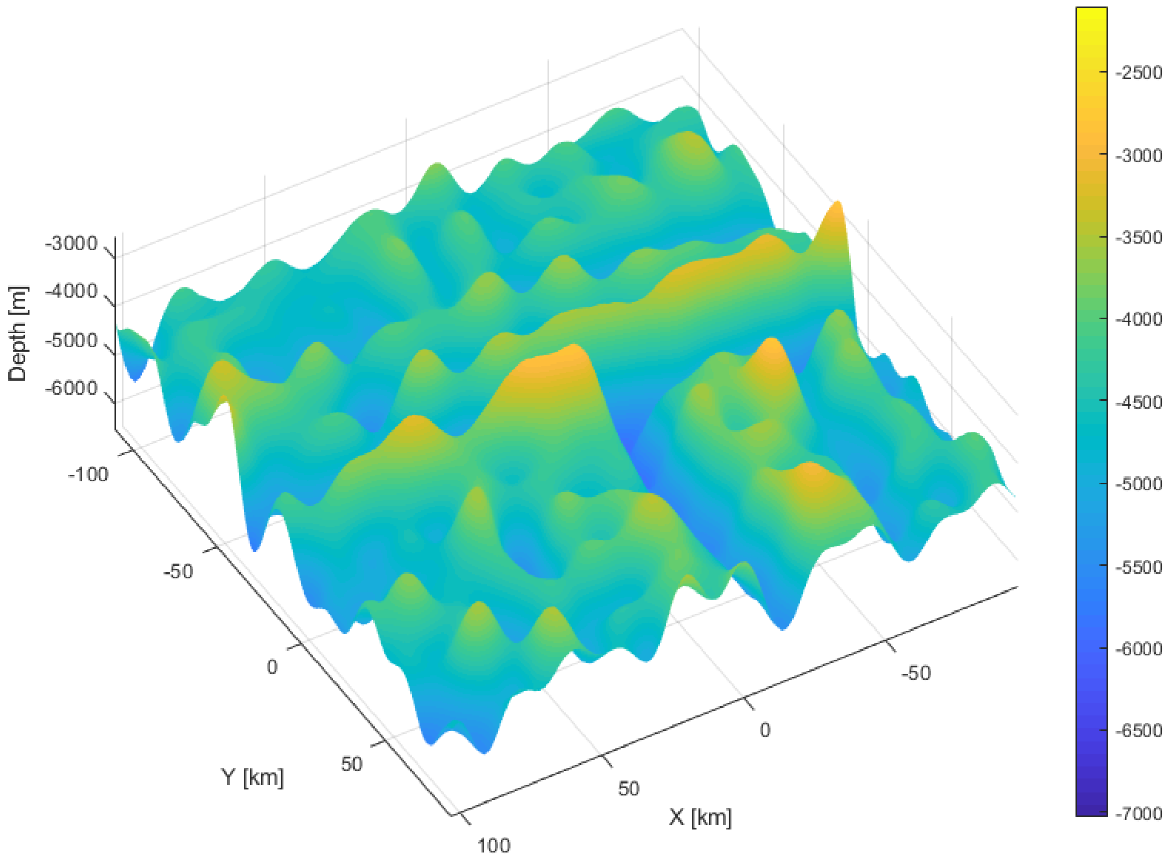


Figure 4.16: PSO estimated topography, 3000 RBFs, RMSE ≈ 17%

BIBLIOGRAPHY

[1] Junjun Yang, Christopher Jekeli, and Lintao Liu. Seafloor Topography Estimation From Gravity Gradients Using Simulated Annealing. *Journal of Geophysical Research: Solid Earth*, 123(8):6958–6975, aug 2018.

[2] Walter H. F. Smith and David T. Sandwell. Conventional bathymetry, bathymetry from space, and geodetic altimetry. *Oceanography*, 17(1):8–23, 2004.

[3] David T. Sandwell and Walter H.F. Smith. Marine gravity anomaly from Geosat and ERS 1 satellite altimetry. *Journal of Geophysical Research B: Solid Earth*, 102(B5):10039–10054, 1997.

[4] Junjun Yang. *Seafloor topography estimation from gravity gradients*. PhD thesis, The Ohio State University, 2017.

[5] Walter H. F. Smith and Karen M. Marks. Seafloor in the malaysia airlines flight mh370 search area. *EOS, Transactions American Geophysical Union*, 95(21):173–174, 2014.

[6] R. L. Parker. The Rapid Calculation of Potential Anomalies. *Geophysical Journal of the Royal Astronomical Society*, 31(4):447–455, 1973.

[7] Soumya Mohanty. *Swarm Intelligence Methods for Statistical Regression*. Chapman and Hall/CRC, 2018.

APPENDIX A

APPENDIX A

CODE MANUAL

In this appendix I will present the names of MATLAB scripts and explain their use along with their input/output parameters.

1.1 MATLAB entry scripts

- The **maintoposcript** is the main and only script that is needed to be run by the user.
  INPUT variables are:
  - *NumGauss* - the number of gaussians used in the generation of the topography.
  - *NumTerm* - the number of terms used in the Parker series approximation when computing the VGG from topography in the fitness function
  - *Xsize* and *Ysize* - sizes of the VGG data matrix along the x and y directions respectively
  - *sigMin* and *sigMax* - minimal and maximal values for the standard deviation of the RBFs that will be used to generate the design matrices.
  - *xMin*, *xMax*, *yMin* and *yMax* are variables denoting the ranges of x and y dimensions of the search space.

  OUTPUT of the script is the topography that best fits the VGG data and three plots: the true topography, the topography generated by PSO and the plot of the RMS error distribution. It also displays the number of RBF's used since this is the main parameter that determines the time of computation.

- **crcbpso** is the main lbest PSO function. It uses the following (mostly standard and recommended in [7]) settings:

Setting	Value/Description
Position initialization	$x_j^{(i)}[0] \to U(x;0,1)$
Velocity initialization	$v_j^{(i)}[0] \to U(x;0,1) - x_j^{(i)}[0]$
$v_m ax$	0.5
N_{part}	40
$c_1 = c_2$	2.0
$\omega[k]$	Linear decay from 0.9 to 0.4
Boundary condition	Let them fly (see Chapter 3)
Termination condition	200 iterations
Topology	Ring topology with neighborhood size of 3

Table 2. PSO settings

1.2 Functions called by the main script

- **datatopogenerator** is the function that generates the topography from the data provided by the user. The data should be in a format of a 2D matrix of arbitrary size. For instance, the data used in this work was in a format of a 3 x 15000 matrix.

  INPUT is the intended X dimension size of the topography and the data file itself.

  OUTPUT is the data matrix of the size: Xsize x Ysize, where Ysize is computed by the function

- **dataVGGgenerator** is the function that generates VGG data from the data provided by the user.

  INPUT is the intended X dimension size of the 2D VGG map at the sea level and the data file itself.

  OUTPUT is the VGG data matrix of the size: Xsize x Ysize, where Ysize is computed by the function

- **heightMin** is the function that calculates the optimal heights of gaussians analytically.

  INPUT:

  VGGdata - VGG data matrix,

  Sigma - array of standard deviations,

  Xmean, Ymean - Array of X and Y coordinates of the centers of RBFs

  Xsize, Ysize - sizes of X and Y dimensions of the VGG/topography

  OUTPUT:

  hminvec - the array of optimal heights

- **fitfun** is the fitness function that is called by the handle in the main script (maintoposcript). The fitness function calculates the heights analytically using the heightMin function described above and computes VGG for a design topography using VGGcomp function to be described below.

  INPUT:

  Sigma - same as above

  Xmean,Ymean - same as above

  vggparams - a structure (VGG parameters), created in the mainscript and containing fields for: Xsize, Ysize, NumTerm, NumGauss, VGGdata. Each one of them is described above.

  OUTPUT:

  fitness - The fitness value of the input VGG data.

1.3 Auxiliary functions

This section briefly describes auxiliary functions used in the code.

- *constVGG* - stores the constants used in the code - made for encapsulation purposes.

- *gaussian unit h* - produces 2D RBFs with height h=1. Used for generating topography from RBFs.

- *RadBasFun* - produces the RBF with a given height h.

- *sum gaussians* - produces the topography from separate RBFs
- *RootMeanSq* - computes the root mean square error normalized by the mean height of the given topography.
- *XYmatfromdata* - produces a matrix of proper dimensions from input data.

BIOGRAPHICAL SKETCH

Yelbir Kazhykarim was born in 1987 in Karaganda city, Kazakhstan. He started his first degree of a Specialist in Aerospace Engineering in Moscow Aviation Institute in 2005 and graduated in 2011. Afterwards he worked for 4 years, first as Physics teacher in Nazarbayev Intellectual School in Karaganda and then as a Research Assistant in the Physics Laboratory of the Center for Energy Research in Nazarbayev University. While working in Nazarbayev University he also obtained an MS in Educational Leadership from Nazarbayev University (2013-2014) and then enrolled in an MS program at UTRGV in 2015. He first received MSIS in Science and Technology in 2017 and then received MS Physics degree in 2020 from UTRGV. His current (until August 2020) address is: Apt. 6B, 1900 West University Blvd, Brownsville, TX, 78520. His permanent mailing address is: Apt. 210, Republic prospect 4, Karaganda city, Kazakhstan, 100025. Email: yelbir.kazhykarim@gmail.com