2-D Inversions of Gravity Data for Multi-Sided Polygons using Particle Swarm Optimization.

by

© Scott Cranford

A Thesis submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science (Geophysics)

Department of Earth Science

Memorial University of Newfoundland

October 2014

St. Johns Newfoundland

ABSTRACT

Since its introduction in 1995 particle swarm optimization (PSO) has been an area of great interest for many optimization problems, including geophysical inversions, as it does not use gradient information to find solutions. PSO is a global searching method, which, in principle, should avoid local minima, and find the global minima. PSO operates by moving a group of particles around the search space according to a simple mathematical formula which is dependent on particle positions and velocities. The particles movements are influenced by their own best known positions, *pbest*, and the overall best known position, *gbest*, which is updated each iteration. This thesis presents a method for inverting 2-D geophysical data using multisided polygons and PSO which has never been investigated in the literature. The first implementation of the algorithm presented below was designed to minimize several standard benchmark functions. The second implementation augmented the existing algorithm by including a forward modeling method, and model parameterization to invert 2-D gravity data. Investigations were then carried out to discover more robust methods of parameterization and constraint handling methods because unconstrained inversions tend to turn themselves inside-out, i.e., become infeasible or geologically implausible. A penalty function was found to be the best solution for constraint handling, with an appropriate penalty value. The algorithm was tested with ten-sided models and noisy data, and returned excellent inversion results for all obstacles presented to it.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Colin G. Farquharson for all of his immeasurable aid and guidance during the course of this thesis. Without him it would not have been possible. Secondly, I would like to thank Dr. Charles Hurich for his willingness to answer any questions that arose as my research progressed. Finally, I would like to thank The Special Scholarships for Students to Pursue Graduate Studies Related to Resource Development, Buchans Scholarship Fund of ASARCO Incorporated and The School of Graduate Studies F. A. Aldrich Fellowship for the funding they provided throughout the course of my graduate program.

Table of Contents

ABSTRACT ii

ACKNOWLEDGEMENTS iii

Table of Contents iv

List of Tables viii

List of Figures ix

List of Appendices xvii

Chapter 1 : Introduction 1

Chapter 2 : Introduction to PSO 5

Chapter 3 : PSO for function minimization 17

3.1 Coding a minimization function 17

3.2 Testing PSO minimization performance on benchmark functions. 19

3.2.1 The First function of De Jong aka, the Sphere function 20

3.2.2 The Rosenbrock function 21

3.2.3 Rastrigins function 23

3.2.4 The Griewangk function 24

3.2.5 The Schaffer F6 Function 26

3.3 Minimization results and comparison 27

3.3.1 Minimization results for the sphere function. 28

3.3.2 Minimization results for the Rosenbrock function. 29

3.3.3 Minimization results for Rastrigins function. 30

3.3.4 Minimization results for the Griewangk function.....32

3.3.5 Minimization results for the Schaffer F6 function.....33

3.4 Conclusions.....34

Chapter 4 : Sides Inversion Method..... 36

4.1 Forward modeling theory.....36

4.2 Converting function minimization code to an inversion algorithm 38

4.2.1 Inversion targets.....38

4.2.2 Model parameterization. 40

4.2.3 Synthetic data and data misfit function.....43

4.2.4 Constraint handling methods 45

4.3 Results.....47

4.3.1 Unconstrained four-sided inversions 48

4.3.2 Unconstrained five-sided inversions.....61

4.3.3 Constrained four-sided inversions 70

4.3.4 Constrained five-sided inversions.....78

4.4 Conclusions.....91

Chapter 5 : Constraint Handling and Points Methods 94

5.1 Forward modeling methods and misfit function. 94

5.2 Model parameterization and inversion targets. 96

5.3 Constraint handling methods and required changes. 101

5.4 Results for five-side inversions with penalties, and six-sided points method. 103

5.4.1 Inversions of five-sided models with a penalty function value of 500 103

5.4.2 Inversions of five-sided models with a penalty function value of 50,000 114

5.4.3 Inversions of five-sided models with RIS method. 121

5.4.4 Inversion Results of six-sided model with points method of parameterization. 134

5.4.5 Advantages Parallel Python. 140

5.5 Conclusions. 148

Chapter 6 : Further Testing of Points Method 150

6.1 Introduction 150

6.2 Ten-sided inversion target, and noisy data creation. 151

6.3 Results 154

6.3.1 Results for ten-sided inversions of noise free data. 154

6.3.2 Results for ten-sided inversions using medium noise. 156

6.3.3 Inversion time differences 160

6.4 Conclusions 161

Chapter 7 : Conclusions 163

Bibliography 165

Appendix 167

List of Tables

Table 4.1: Inversion results of tests performed on four-sided models without constraints. The columns are as follows: Test # is the number order in which the tests were performed, misfit is the misfit achieved by the inversion, X1 is the left side coordinate, X2 is the right side coordinate, Z1 is the coordinate of the top and Z2 is the coordinate of the bottom. 56

Table 4.2: Inversion results of tests performed on five-sided models without constraints. 65

Table 4.3: Inversion results of tests performed on four-sided models with corner switching constraints. 76

Table 4.4: Inversion results of tests performed on five-sided models with Corner switching constraints. 79

Table 5.1: Inversion results for constrained five-sided example using a penalty function value of 500. 106

Table 5.2: Results obtained from the five-sided constrained inversions with sides method using a penalty value of 50,000. 117

Table 5.3: Results obtained from inversions with sides method using the RIS method and 10,000 iterations. 132

Table 5.4: Results obtained for inversions using the points method, with penalty value 50,000 on a six-sided model. 145

List of Figures

Figure 1.1: Target and predicted models, with synthetic and predicted data.....1

Figure 2.1: Simplified model of PSO method for minimization or inversion (Yuan et al., 2009). 7

Figure 2.2: Graph of median evaluations (cost) vs. population size (Carlisle and Dozier, 2001) 11

Figure 3.1: Python plot showing the sphere function between -5.12 and 5.12 with the global minimum represented by the yellow dot.....21

Figure 3.2: Python plot showing the Rosenbrock function between -2.048 and 2.048 with the global minimum represented by the yellow dot.....22

Figure 3.3: Python plot showing Rastrigins function between -5.12 and 5.12 with the global minimum represented by the yellow dot.....23

Figure 3.4: Python plot showing the Griewangk function between -600 and 600 with the global minimum represented by the yellow dot.....25

Figure 3.5: Python plot showing the Griewangk function between -5.12 and 5.12 with the global minimum represented by the yellow dot.....25

Figure 3.6: Python plot showing the top view of the Schaffer F6 function between -20 and 20 with the global minimum represented by the yellow dot.....26

Figure 3.7: Python plot showing the Schaffer F6 function between -5.12 and 5.12 with the global minimum represented by the yellow dot.....27

Figure 3.8: Plot of 2D sphere and an example convergence path.....28

Figure 3.9: Plot of the Rosenbrock function and an example convergence path.....30

Figure 3.10: Plot of the Rastrigins function and an example convergence path.....31

Figure 3.11: Plot of the Griewangk function and an example convergence path. 32

Figure 3.12: Plot of the Schaffer F6 function and an example convergence path. 34

Figure 4.4.1: Approximation of cross sectional shape using a polygon (Blakely, 1996). .37

Figure 4.4.2: Initial inversion target (blue) as well as boundaries of the survey region (black box), with the observation locations (red triangle). 39

Figure 4.4.3: Second inversion target considered in this chapter, five sides (blue).....40

Figure 4.4.4: Diagram showing how parameterization of four-sided objects, using the sides method, is performed. 41

Figure 4.5: Diagram showing how parameterization of five-sided objects, using the sides method, is performed. 42

Figure 4.6: Diagram of four-sided model with its synthetic data profile.....43

Figure 4.7: Diagram of five-sided model and its synthetic data. 44

Figure 4.8: Depiction of a four-sided model which has become infeasible by turning inside out, and the predicted data it would generate plotted with the synthetic data.....47

Figure 4.9: Line plot of inversion results returned from tests on four-sided model without constraints. 49

Figure 4.10: Diagram of predicted model from inversion result with misfit 226, with its predicted data (red+), shown with the target model and the synthetic data (blue). 50

Figure 4.11: Diagram of predicted model from inversion result with misfit closest to average value, with its predicted data (red+), shown with the target model and the synthetic data (blue). 51

Figure 4.12: Diagram of predicted model from inversion result with lowest misfit value of this set, and its predicted data (red +), shown with the target model and the synthetic data (blue). 52

Figure 4.13: Diagram of all model shapes returned for unconstrained four sided inversions. 54

Figure 4.14: Diagram of target model (green corners and blue lines), with the eight models with lowest misfit in this suite. 58

Figure 4.15: Diagram of target model (green corners and blue lines), with the eight models with lowest misfit in this suite. 59

Figure 4.16: Diagram of target model (green corners and blue lines), with four highest misfit models and four middle misfit models. 60

Figure 4.17: Line plot of inversion results returned from tests on five-sided model without constraints. 62

Figure 4.18: Diagram of the predicted model and data (red +), for the highest inversion result with misfit 189, shown with target model and synthetic data. 64

Figure 4.19: Model and predicted data result for test number 6 (red +), with synthetic data. 67

Figure 4.20: Model and predicted data result for test number 12 (red +), with synthetic data. 68

Figure 4.21: Model and predicted data result for test number 20 (red +), with synthetic data. 69

Figure 4.22: Line plot of inversion results returned from tests on four-sided model with corner switching constraints. 70

Figure 4.23: Model and predicted data result for test with misfit value 86.0 (red +), with synthetic data (blue). 71

Figure 4.24: Data difference plot for model 4, with misfit 86.0. 72

Figure 4.25: Data difference plot for model 12, with misfit 19.5. 73

Figure 4.26: Data difference plot for lowest misfit model returned, with misfit 0.1. 74

Figure 4.27: Plot of all model shapes returned from inversion of five-sided inversion with corner switching constraints. 81

Figure 4.28: Line plot of inversion results returned from tests on five-sided model with corner switching constraints. 82

Figure 4.29: Observed (blue) and predicted (red+) data for model 16, with misfit 607.55. 84

Figure 4.30: Data difference plot for models 16, 13, and 10 with misfit values 607.55, 492, and 432.31 respectively. 85

Figure 4.31: Model shapes for test numbers 16, 13 and 10, with target model. 86

Figure 4.32: Observed (blue) and predicted (red+) data for model 6, with misfit 143.80. 88

Figure 4.33: Model shape for test number 6, with target model. 89

Figure 4.34: Data difference plot for model 6, misfit 143. 89

Figure 4.35: Model shapes for test numbers 11, 7 and 5, with target model. 90

Figure 4.36: Data difference plot for models 11, 7, and 5 with misfit values 1.12, 1.6, and 3.08 respectively. 90

Figure 5.1: Plot of two identical sets of synthetic data calculated using code for the sides method and points method. 95

Figure 5.2: Plot of five possible shapes created by the new initialization procedure used for the points method. 97

Figure 5.3: Depiction of simple and complex polygons: simple represents a feasible model and complex a non-feasible model (Pierce 2013). 98

Figure 5.4: Demonstration of how the exterior angles of a polygon add to 360 degrees (Pierce, 2013). 99

Figure 5.5: Example of a complex polygon with exterior angles adding to 360 degrees (John, 2013). 99

Figure 5.6: Depiction of the six-sided model and synthetic data from the 4,5 and 6 sided models, represented in light blue, red and dark blue respectively. 101

Figure 5.7: Plot of inversion results and average objective function value from constrained five-sided tests using a penalty function with a value of 500. 105

Figure 5.8: Plot of the model returned in test inversion 10, with predicted and synthetic data indicated by red crosses and the blue line respectively. 108

Figure 5.9: Plot of the models returned in test inversions 3 and 8 (green and red dots and lines respectively), with predicted and synthetic data (green and red crosses, and blue line respectively). 110

Figure 5.10: Predicted data (red+) and synthetic data (blue line) for run 7 of five-sided example constrained with a penalty value of 500. 111

Figure 5.11: Predicted data (red+) with synthetic data (blue line) for run 9 of five-sided example constrained with a penalty value of 500. 111

Figure 5.12: Plot of model shapes returned from test inversions 7 and 9 (and the true model) for the five-sided examples constrained using a penalty function value of 500.. 112

Figure 5.13: Predicted data (red+) and synthetic data (blue line) for run 19 of five-sided example constrained with a penalty value of 500. 113

Figure 5.14: Predicted data (red+) and synthetic data (blue line) for run 20 of five-sided example constrained with a penalty value of 500. 114

Figure 5.15: Line plot of objective function values obtained from inversions using a penalty value of 50,000. 116

Figure 5.16: Objective function value omitting the outlier of 28,013 116

Figure 5.17: Predicted data (red crosses) and model for run 20 using penalty value 50,000. 119

Figure 5.18: Predicted data (red crosses) and model for run 16 using a penalty of 50,000. 120

Figure 5.19: Set 1 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations. 122

Figure 5.20: Set 2 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations. 122

Figure 5.21: Set 3 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations. 123

Figure 5.22: Plot of predicted data of model 7 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 7 is 591. 125

Figure 5.23: Plot of predicted data of model 14 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 14 is 986. 126

Figure 5.24: Plot of predicted data of model 8 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 8 is 13912. 127

Figure 5.25: Data difference plot for predicted data for models 7, 14, 19, and 12 from the RIS, five-sided constrained inversions. 127

Figure 5.26: Plot of constrained models for each of the three subsets of results. 129

Figure 5.27: Inversion results and average misfit values of results from RIS method experiments with 10,000 iterations, outliers included. 130

Figure 5.28: Inversion results and average misfit values of results from RIS method experiments with 10,000 iterations, outliers removed. 131

Figure 5.29: Line plot of inversion results and average misfit values from six-sided points method experiments. 136

Figure 5.30: Plot of predicted data (green line) for six-sided inversion model 11 (which has a misfit 88.52); observed data are shown by the blue line. 137

Figure 5.31: Plot of model shapes returned from tests with points method and penalty function for the six-sided examples. 139

Figure 5.32: Data difference plots for models 11, 16, and 18 from tests of points method on six-sided targets. 139

Figure 6.1: Ten-sided inversion target (blue) inside the survey region, which is outlined in black. 151

Figure 6.2: Diagram of the six- and ten-sided models, with their synthetic data plotted above. Six-sided model is in red, with pink gravity curve. Ten-sided model is blue with blue gravity curve. 152

Figure 6.3: Synthetic data plots for no noise (smooth blue), low noise (red), medium noise (jagged blue) and high noise (yellow). 153

Figure 6.4: Line plot of noise-free 10 sided inversion results (blue) with average misfit value (red). 154

Figure 6.5: Plot of predicted data(green) for run 2 (which has a misfit of 80.71) of the noise-free ten-sided inversions. The blue curve shows the noise-free observed data..... 155

Figure 6.6: Plot of the noise-free synthetic data (blue) with the medium noise data (yellow). 156

Figure 6.7: Plot of predicted data for model 11, (misfit 956.51) (green), with noise free (blue) and medium noise (yellow) data sets..... 157

Figure 6.8: Plot of predicted data for model 1, which has a misfit 1103.16 (green), with noise free (blue) and medium noise (yellow) data sets. 159

Figure 6.9: Diagram of models 1 and 11, with base model for reference..... 160

List of Appendices

Appendix A…….165

Appendix B…….Disc

Chapter 1 : Introduction

Solid earth geophysical studies generally aspire to learn something about the interior of the earth from measurements of physical quantities taken at or above the earths surface Blakely (1996). The type of measurement that will be considered during the course of this research is the anomalous gravitation field caused by a buried object. There are several methods in which objects or structures in the subsurface can be deduced using data collected at the surface. Two methods in particular of interest in this case are the forward modeling method and the inversion method. The forward method is essentially a three step process which can be explained using Figure 1.1.

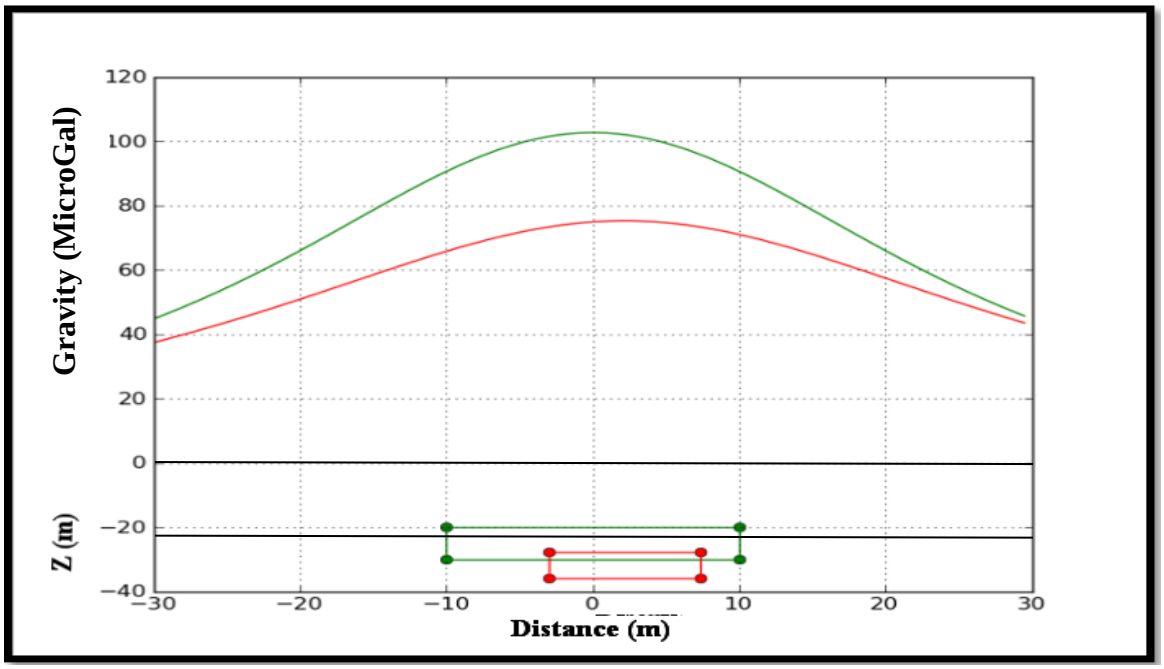

Figure 1.1: Target and predicted models, with synthetic and predicted data.

If the green rectangle is the true buried body, then the green curve is the anomalous field associated with it. The three steps to the forward modeling method are as follows: first create an initial model for the source body based on geologic and geophysical information; this is represented by the buried red rectangle in Figure 1.1. Next calculate the anomaly associated with the predicted model and compare it to the observed anomaly; the calculated anomaly is the red curve, which when compared to the green observed anomaly requires some improving. Lastly, the model parameters, such as the length, height, position, and density, are adjusted in order to improve the fit between the two anomalies. These three steps are repeated until the difference between the curves of the two anomalies is deemed acceptable (Blakely, 1996).

Unlike the forward modeling method, where the source parameters are estimated indirectly through trial and error calculations, the inverse method determines the source parameters in a more direct way from the gravity measurements. There are two approaches to the inverse method, the first is known as linear inversion and the second is known as non-linear inversion. Since the gravitational field is linearly dependant on the density of an object, linear inversion problems are those that involve discretizing the survey region into small cells and changing the density value in each cell to fit a data set (Li and Oldenburg, 1996). The non-linear inversion method on the other hand attempts to estimate the parameters that define a source, such as its depth, its dimensions and its density, which created an observed anomaly (Blakely, 1996). These are called non-linear inversions because multiple parameter can be adjusted in every iteration. According to Yaun et al. (2009) most practical geophysical inversion problems are non-linear, so

naturally we turn to non-linear inversion methods to solve these geophysical problems. In general there are two types of non-linear inversion methods: (1) local search techniques and (2) global search techniques. Global search techniques developed rapidly over the past 20 years with notable methods such as the Monte Carlo method, simulated annealing (SA), genetic algorithm (GA), and many more (Yuan et al., 2009). In these techniques, searches are mainly applied to obtain satisfying solutions by iteration. The SA and GA, algorithms and improved heuristics global search techniques based on SA and GA (those which trade optimality, or precision for speed), are widely researched and applied. However the downside to these methods may include slower convergence speed. Currently, in the interdisciplinary optimization field, swarm intelligence is the new 'IT' topic. These mainly consist of the Ant Colony optimization (ACO) algorithm, and Particle Swarm Optimization (PSO), which is simple and easy to implement and there is no need to adjust many parameters. The PSO algorithm is the focus of this thesis. It was first introduced by Kennedy and Eberhart in 1995 and was designed to mimic birds as they search for food. To do this it consists of two memory components, one that drives each particle to search near its best known location so far found, and one that drives it towards the best overall location found by the entire swarm. In this way PSO has the merits of a local search technique as well as those of a global search technique.

The purpose of this thesis is to create an inversion algorithm using PSO, and to test its effectiveness inverting geophysical data for multi-sided polygonal shapes. Each step in the process of creating the code will be presented from optimizing standard test functions to inverting for multisided polygons. This work will demonstrate that using

PSO for geophysical inversions is a viable option for conducting non-linear inversions for complex objects. Chapter 2 will give a thorough understanding of PSO, from its conception to its current uses, as well as its evolution along the way. The following chapter will detail the PSO algorithms use on function optimization during this research, using several benchmark functions and comparison to other research in this area. Chapter 4 will then delve into augmenting the PSO algorithm, using the sides method of parameterization, for use in geophysical inversion, as well as some examples of its success and failure. The next chapter being related to the previous chapter will explain the changes made to the algorithm, describing methods for constraining the inversions. It also introduces a new parameterization technique, which was developed to deal with the failures of the previous implementation. Chapter 6 will then increase the difficulty of the inversions by adding additional sides, and adding noisy data sets. Finally Chapter 7 is an overview of the research presented.

Chapter 2 : Introduction to PSO

In 1995 James Kennedy and Russell Eberhart published “Particle Swarm Optimization” (Kennedy and Eberhart, 1995). Their paper describes a new method for optimizing continuous nonlinear functions known as particle swam optimization (PSO) which was discovered through simulations of a simplified social model, and which eventually evolved into an optimizer. The concept of PSO has its roots in artificial life, and is also related to evolutionary computation. Evolutionary computation techniques work on a population of potential solutions in a search space, and are often based on things observed in nature, such as birds flocking, fish schooling, and swarming theory.

Similar to a genetic algorithm (GA), PSO is initialized with a population of random solutions; however it is much simpler since it dispenses with the need to mimic complex gene transfers and mutation (Eberhart and Kennedy, 1995). In PSO the solutions are known as “particles” which are given a randomized velocity and are “flown” through hyperspace, which essentially means they move around in the search region. The particles are continuously searching for better solutions. Unlike the gene transfers and analogs of thermodynamic processes which are found in GA and simulated annealing (SA), PSO has a memory component for each particle in the swarm so both the “cognitive knowledge” (each particles individual knowledge) and “social behavior” (knowledge of the swarm as a whole) of the particles is used to guide the movement of the particles through the search space. In this way, PSO contains merits of both local and global search algorithms (Shaw and Srivastasa, 2007). Each particle is treated as a point in a D-dimensional space where

the ith particle is represented as $\mathbf{X_i = (x_{i1} ,.x_{i2},.....,x_{iD})}$. As the particles traverse the search region, each individual keeps track of its current coordinates and its best known solution (fitness or misfit) that has been achieved so far. This best solution is called *pbest*, and is represented as $\mathbf{P_i= (P_{i1},P_{i2}, . . . , P_{iD})}$. The particles also track the best overall position and its associated value of fitness or misfit found, which is known as the global best, or *gbest* represented as $\mathbf{P_g= (P_{g1},P_{g2}, . . . , P_{gD})}$, where D is the number of dimensions used. This can be found by any individual, and its information is shared with every particle in the swarm.

In the early versions of PSO the concept consisted of the following: at each time step change the velocity of each particle towards its *pbest* and the *gbest*, where the velocity for particle ***i*** is represented as $\mathbf{V_i = (V_{il},V_{i2}, . .. , V_{iD})}$. To better understand the way in which PSO works, a simplified example of how PSO would minimize a problem or inversion is presented in Figure 2.1

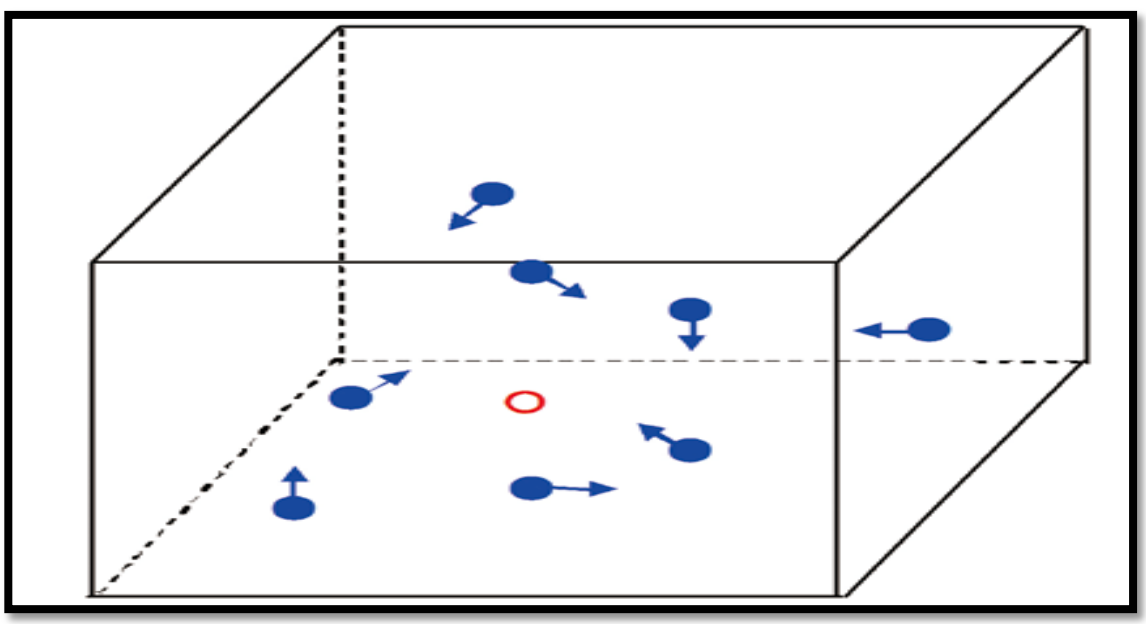

Figure 2.1: Simplified model of PSO method for minimization or inversion (Yuan et al., 2009).

In this example, to be clearer, only three dimensional space is considered instead of D-dimensional space. The cube represents the available search space, the blue circles are particles, the arrows are the velocity vectors and hence the direction in which the particles will move at this iteration, and let us assume that the red open circle is the known best global result. Each particle will move a distance determined by its velocity each iteration. A new velocity is calculated every iteration, and is based on 1) the previous velocity, 2) the difference between a particles current position and its own best position, and 3) the difference between a particles current position and the best known overall position. In this way particles are continuously moving towards, or searching for, better solutions, until they find the global best solution. The original equation for calculating velocity is defined as (Carlisle and Dozier, 2001):

$$\mathbf{V_i = V_i + \varphi_1 * rand(\,) *(P_i - X_i) + \varphi_2 * Rand(\,) * (P_g - X_i)} \qquad (1)$$

$$\mathbf{X_i = X_i + V_i} \text{ (a time step of one)} \qquad (2)$$

where $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$ are positive constants, and rand( ) and Rand( ) are two random functions in the range [0,1] and * is standard multiplication. These random numbers make PSO less predictable and therefore become more susceptible to becoming trapped in local minima. The second set of terms in Equation 1 is known as the cognitive component which represents the movement of each individual particle. The third part of the equation is known as the social component, which represents the interaction between the particles (Shi and Eberhart, 1999). Constants $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$ determine the relative importance of the

cognitive and social components respectively. In Equation 2 the particle moves towards its new position with the velocity calculated in Equation 1.

There have since been several modifications of the algorithm, beginning with Shi and Eberhart (1999). These authors introduce a variation of PSO by adding an *inertia weight factor (W).* This value is considered crucial for PSOs convergence behavior. They show that PSO searches wide areas effectively but it lacks local search precision. The inertia factor was introduced to try and adjust the velocity over time to correct this behavior. In the beginning of the search the inertia factor is large to facilitate global exploration (searching new areas). It decreases throughout the search to facilitate fine tuned local exploration. Adding this inertia factor changes Equation 1 to:

$$\mathbf{V_i = W * V_i + \varphi_1 * rand( ) * (P_i - X_i) + \varphi_2 * Rand( ) * (P_g - X_i)} \qquad (3)$$

while Equation 2 is unchanged. Selecting an inertia weight that is constant and is also near the middle of the acceptable values described will give a balance between global and local searching abilities.

In Clerc (1999) the inertia weight factor is removed in favor of a new term called the *constriction factor* (K). K is simply a different way to control and constrain velocities. K dynamically lowers the velocities as time progresses, which focuses on a local search in later iterations, an idea introduced by Shi and Eberhart (1999). The constriction factor is defined as:

$$K = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|} \qquad (4)$$

where $\varphi = \varphi_1 + \varphi_2$, and $\varphi > 4$. The constriction factor K changes Equation 1 to:

$$\mathbf{V_i = K (V_i + \varphi_1 * rand( ) * (P_i - X_i) + \varphi_2 * Rand( ) * (P_g - X_i))} \ . \qquad (5)$$

Again Equation 2 is unchanged. Lakari et al. (2002) investigated the convergence of PSO using the inertia factor, Clercs constriction factor and a variant that uses both together. They conducted experiments on seven widely used test problems and found that the variant using both W and K converged fastest, but using only W or only K are better for global convergence, especially in higher dimensional problems. In most of the experiments, PSO using only Clercs constriction factor was significantly faster than using only the inertia factor. It is for this reason, and the superior global searching, that this thesis uses only Clercs constriction factor and not the inertial weight factor, or a combination of both factors.

Since the introduction of PSO each different researcher seems to have their own preferred implementation, using different population sizes, different neighborhood sizes, etc. Carlisle and Dozier (2001) investigated this by examining a variety of these choices in an effort to determine a "canonical particle swarm optimization" which would act as a good starting point for applying PSO in their paper "An off the shelf PSO". To test the various parameters they started off by using the same parameters used by Shi and Eberhart (1999). The population size was set to 30, constants $\boldsymbol{\varphi_1}$ and $\boldsymbol{\varphi_2}$ were both set to 2.05, V_{max}, which is the maximum velocity a particle can achieve at each iteration, was set to X_{max}, the maximum distance a particle can move at each iteration, and they incorporated Clercs constriction factor. Since they also wished to test neighborhood size and methods for updating the particles, which had never been tested before they made the assumptions that the neighborhood would be global, and the particles would update

synchronously (i.e. *gbest* would be determined between iterations). To gain knowledge about how these 6 parameters could be improved, or not, Carlisle and Dozier came up with a systematic approach for evaluating how each change affected the convergence. Firstly they would only change one parameter at a time to gauge the effect on the convergence. Secondly they would test each change in parameter on five different benchmark functions to determine if a change was an improvement in a general sense and not just an improvement for that individual case. Thirdly they performed every experiment on each function twenty times with an upper limit of 100,000 iterations. To measure how the changes affected the outcome they recorded the success rate, the average and median iterations for successful runs, and the average and median counts of calls to the evaluation function. The latter is a measure of the actual work performed by the algorithm within the set of runs (Carlisle and Dozier, 2001).

The first experiment Carlisle and Dozier carried out was designed to indentify the effects of changing the population size. Shi and Eberhart (1999) reported that PSO was not sensitive to population size and Carlisle and Dozier (2001) found that this was generally true in terms of performance but not cost. They varied the population size from 5 to 200 particles with an interval of 5 evaluating with the above mentioned criteria. The expected conclusion was that with more particles more search space is covered and the solution would be found sooner since there are more guesses at the solution per iteration and more shared knowledge throughout the swarm. This was true in terms of the number of iterations it took to find the solution. However, as the population increases, each iteration represents a greater cost, as more particles call upon the evaluation function.

Carlisle and Dozier (2001) report that the increase in cost more than offsets the reduction in iterations seen in Figure 2.2, a graph of median evaluations (cost) vs. population size. They therefore conclude that a population size of 30 particles is small enough to be efficient while still producing reliable results.

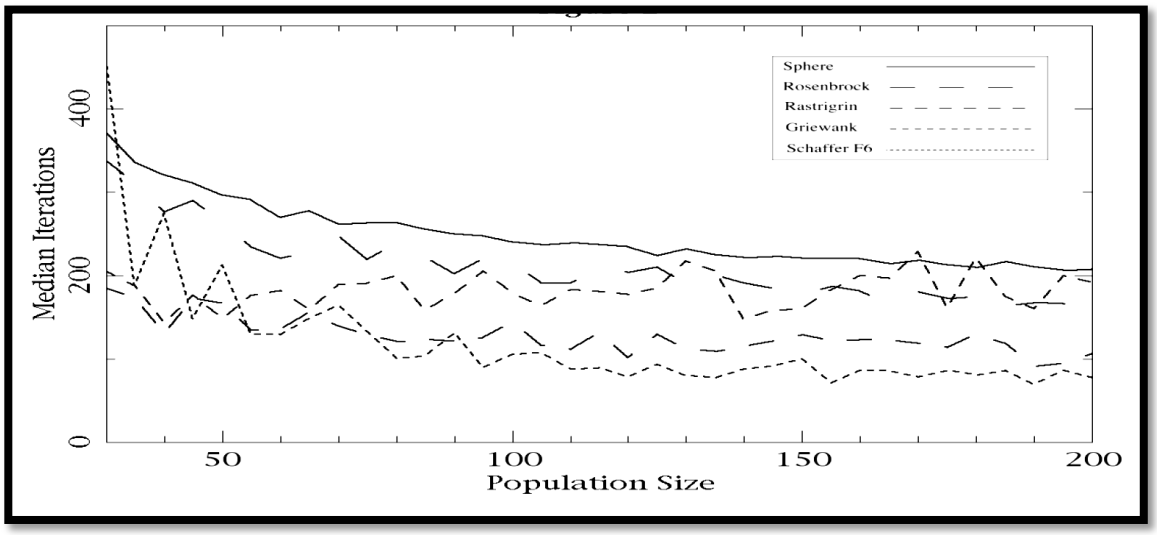

Figure 2.2: Graph of median evaluations (cost) vs. population size (Carlisle and Dozier, 2001)

The neighborhood size for a swarm is the number of neighboring particles that influence a particular particles movement (Carlisle and Dozier, 2001). According to Eberhart and Kennedy (1995), small neighborhoods are better at avoiding local minima, and global neighborhoods converge faster. For their experiment Carlisle and Dozier (2001) varied the size of the neighborhood from 2 to 30 in steps of 2. They found that increasing the size does no harm to the convergence in 4 of the 5 test functions and in some cases significantly improved the convergence performance. Therefore they

concluded that the global neighborhood appears to be a better general choice, as it seems to require less work for the same results.

The third set of experiments dealt with synchronous and asynchronous updating of the particles. It seems natural that, in a local neighborhood situation, the local best (*lbest*) is computed asynchronously, that is, as each particle is about to be moved the best neighbor is determined and that influence is applied to the particles motion. On the other hand, when the neighborhood is global, it also seems natural that the evaluation of the global best (*gbest*) is done synchronously, between iterations, so that all particles are moved in parallel, then the best particle in the population is selected as the *gbest*, and the next iteration is run (Carlisle and Dozier, 2001). Apart from the obvious difference between these two ways of updating the particles it must also be pointed out that in synchronous updating all particles move before the best selection is made, and in asynchronous updating the neighbors on one side of a given particle have been updated but the ones on the other side have not. To investigate this difference Carlisle and Dozier ran PSO with various neighborhood sizes and using both types of updating. They report that synchronous updates where almost always more costly with any neighborhood size and therefore asynchronous updating is the preferred choice.

The next area Carlisle and Dozier investigated was the ratio between the cognitive and social factors, $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi_2}$ respectively. They varied cognitive rate from 0.0 to 4.1 and calculated the social value as $\boldsymbol{\varphi_2 = 4.1 - \varphi_1}$. The results of this showed that the usual values of 2.05 and 2.05, which were proposed by Kennedy (1998), did not result in the best performance for any of the test functions. The analysis revealed that with higher social

values the swarm was more likely to be trapped in local minima. Carlisle and Dozier proposed that a good compromise between the cognitive and social ratios is 2.8 and 1.3 respectively.

Once Carlisle and Dozier nailed down a good value for the cognitive and social ratios they then set about investigating the magnitude of $\boldsymbol{\varphi}$**.** Since the previous value of 4.1 was established before the discovery of Clercs constriction factor this experiment was designed to find out if this was still valid or if some new value would work better. They ran tests with values varying from 1.0 to 6.0 while maintaining the ratio found previously. They found that the previously established value of 4.1 was actually the best in all but one scenario, which was later determined to be problem specific. Carlisle and Dozier conclude therefore that leaving the magnitude of $\boldsymbol{\varphi}$ as 4.1 is the best for general cases.

The last experiment Carlisle and Dozier carried out was designed around the V_{max} value. Since Clercs constriction factor should dampen escalating velocities Carlisle and Dozier thought that this value may no longer be needed but decided to investigate because Shi and Eberhart (1999) found that setting $V_{max} = X_{max}$ helps to enforce the X_{max} boundary condition. The experiment used four different V_{max} restrictions, 1) no V_{max}, 2) $V_{max} = X_{max}$, 3) $V_{max} = X_{max}/2$ and 4) $V_{max} = X_{max}/4$. In general they found that performance increases when V_{max} shrinks, but that there must be a lower limit to this behavior since V_{max} is the step size and decreasing it too much would affect the swarms ability to search. They also explored an alternative to fine tuning the V_{max} value where the velocity of a particle is set to zero if it reaches X_{max}. The results are not as good as fine tuning but are quite acceptable since they allow the removal of another parameter, V_{max}.

In light of this Carlisle and Dozier concluded that when X_{max} must be enforced then set the velocity to zero if a particle reaches X_{max}, otherwise V_{max} need not be enforced.

As a final test of the proposed parameters, Carlisle and Dozier tested four sets of previously used parameters against the newly formulated ones. The results showed that besides the large improvement from using Clercs constriction factor the other new parameters had consistent but lesser improvement over other combinations of parameters. Keeping these conclusions in mind the starting parameters used in this thesis are generally the same, namely 30 particles with a global neighborhood. The particles are updated synchronously here despite the above recommendations, since it is much easier to implement. The cognitive-to-social ratio is kept as recommended as well as the magnitude of $\boldsymbol{\varphi}$. Clercs constriction factor, K, is used since this leads to the greatest improvement in performance. There is no constraint put on V_{max}.

Once the starting parameters are set the last thing to consider when using PSO for nonlinear optimization is how to handle the constraints. The uses of constraints will prevent the inversions from violating the boundary conditions of the inversions by leaving the search space. They are also important in controlling the shapes of the models returned. Constraints can be either hard or soft; they can represent physical limits to a search region or simply preferred operating ranges. The rigidity with which these constraints are enforced should reflect the nature of the constraints themselves (Coath and Halgamuge, 2003). Coath and Halgamuge compare two methods for handling constraints: 1) methods based on penalty functions and 2) methods based on the rejection of infeasible solutions. Penalty functions are the traditional means of solving constrained nonlinear optimization

problems. This approach basically allows the removal of all constraints, replacing them with conditions. When one of the conditions is violated, a penalty value is added to the objective function value (when dealing with minimization problems). This leads to optimization of the penalty function and the objective function at the same time. The problem with this method is that a balance needs to be maintained between obtaining feasibility whilst finding optimality (Coath and Halgamuge, 2003). The rejection method does not add penalties when constraints are violated, it simply discards those particle positions. When using this method all particles must be forced into feasible space before any searching can begin. This can be a drawback for this method if the feasible search space is too small. In this method only particles in the feasible region can be considered for *pbest* and *gbest*, the idea being that this will accelerate the iterative process of tracking feasible solutions by forcing the search space to contain only solutions that do not violate any constraints (Coath and Halgamuge, 2003). In order to test which of these methods converges faster and with more accuracy, Coath and Halgamuge used four test functions and recorded the average, best and worst solutions over 500 iterations for each. The tests were carried out under identical swarm parameters, and they found that both of the constraint handling methods had merits. The penalty function method was found to be significantly faster in two of the test cases but the feasibility method found more accurate results on average in 3 out of 4 cases. This means that the choice of method is problem dependant. Coath and Halgamuge also explain that fine tuning the penalty function could lead to increased accuracy. Judging from these results, this thesis will use the penalty

function method because of its faster convergence and comparable optimal solutions found as well as the chance to increase the accuracy of the average solutions.

Chapter 3 : PSO for function minimization

3.1 Coding a minimization function

As mentioned in Chapter 1, the purpose of this thesis is to create an inversion algorithm using PSO, and to test its effectiveness at inverting geophysical data for multi-sided polygonal shapes. The first step taken towards this goal was writing software to implement an algorithm that can minimize 2-Dimensional functions (The source code can be found in Appendix B, along with all results not explicitly shown in this chapter). The results of these minimizations were then compared to similar tests found in the literature, in particular Carlisle and Dozier (2001), to determine if this PSO algorithm is working well enough to move forward and if it is on par with similar PSO code written for the same purpose. The natural thing to do as a beginner programmer, as I was at the start of this venture, was to look for existing PSO code for Python to see if there was something that I could simply adapt to my purpose instead of creating everything from scratch. Unfortunately I didn't find anything that could do the things that I needed, but I did find a few Python scripts that were general PSO algorithms. These scripts showed me how to initialize a swarm and how to move them around in one dimension as well as how to replace previous *pbest* and *gbest* positions and values with newer, better ones found during minimization.

One of the other points that I learned from the existing scripts was to use additional Python packages such as NumPy. NumPy is an extension to the Python programming language that adds support for large, multi-dimensional arrays and

matrices, along with a large library of high-level mathematical functions to operate on these arrays. Using NumPy, it is easy to create “arrays” of numbers, which are ideal for representing the swarm in PSO. These arrays can have different dimensions, which works well for the situation here, since the swarm will have different dimensions for different objectives. In the context of minimizing functions, initializing the swarm was done using an NSwarm by NDim array where NSwarm is the number of particles used and NDim is the number of dimensions in the search space. The number of particles can be changed as needed, but the number of dimensions for the work presented in this chapter is two, which are the x and y coordinates of each particle. An example of a starting swarm of five particles is shown below:

Starting Swarm = [[28.6 28.2][8.8 58.1][13.3 12.6][71.1 18.2] [29.3 69.9]]

Once a swarm has been initialized, the function being minimized must be evaluated for all particles, which is the definition of whichever functions is being minimized. This is also the way in which the *pbest* and *gbest* values are initialized, since it is the first iteration, there are no better areas found because you are evaluating before any movement has occurred. Examples of *pbest* and *gbest* staring arrays can be seen below:

Starting *pbest* = [[17.6 25.6] [74.3 64.1] [39.1 59.9] [15.7 58.4][73.2 21.5]]

Starting *gbest* = [17.6 25.6]

The last piece of the code that needs to be initialized is the velocities of the particles. This is done using a NumPy function called “zeros” which creates an array full of zeros and of the desired shape. An example of starting velocities can be seen below:

Staring velocities = [[0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.]]

The above examples of the swarm, *pbest* and velocities should indicate that all arrays have the same dimensions; the significance of this becomes evident when working with NumPy array manipulation. Arrays of the same shape can be added or subtracted from each other without having to take care of the indexing of every single number being operated on. This makes moving particles a simple matter of adding the swarm and the velocities together. Once all of the necessary pieces of the code are initialized, the swarm, *pbest* and *gbest*, velocities and the objective function defined, the only thing left to start minimizing functions is to iterate over the evaluation and updating of the *pbest* and *gbests* and calculating new velocities using equations (2) and (5), until the global minimum is found.

3.2 Testing PSO minimization performance on benchmark functions.

Once the details of the minimization algorithm had been worked out the next step was to test its performance to make sure that it was operating in the proper manner. Conveniently enough, similar testing of PSO performance on benchmark functions has been done previously by Carlisle and Dozier (2001) as they investigated the best set of starting parameters for an “off the shelf” PSO. Carlisle and Dozier use a set of 5 benchmark functions and minimize these functions 20 times while recording the average number of iterations to find a solution and the success rate of finding a solution. The five

benchmark functions they use are: the first function of De Jong, the Rosenbrock function, Rastirgins function, the Griewangk function and the Schaffer F6 function. The same functions will be used here in order to create similar results for comparison.

3.2.1 The First function of De Jong aka, the Sphere function

The so called first function of De Jong, or the sphere as it is more commonly known as, is one of the simplest test benchmark functions around. The function is continuous, convex and unimodal. It has the following definition in two dimensions:

$$f(\underline{x}) = \sum_{i=1}^{N} x_i^2$$

The test area for this function is generally restricted to a hypercube $-5.12 \leq x,y \leq 5.12$, and has a global minimum of $f(\underline{x}) = 0$ located at $(x_i) = 0$ see Figure 3.1 (Molga and Smutnicki, 2005). The values of N used in this thesis are 2, and 30.

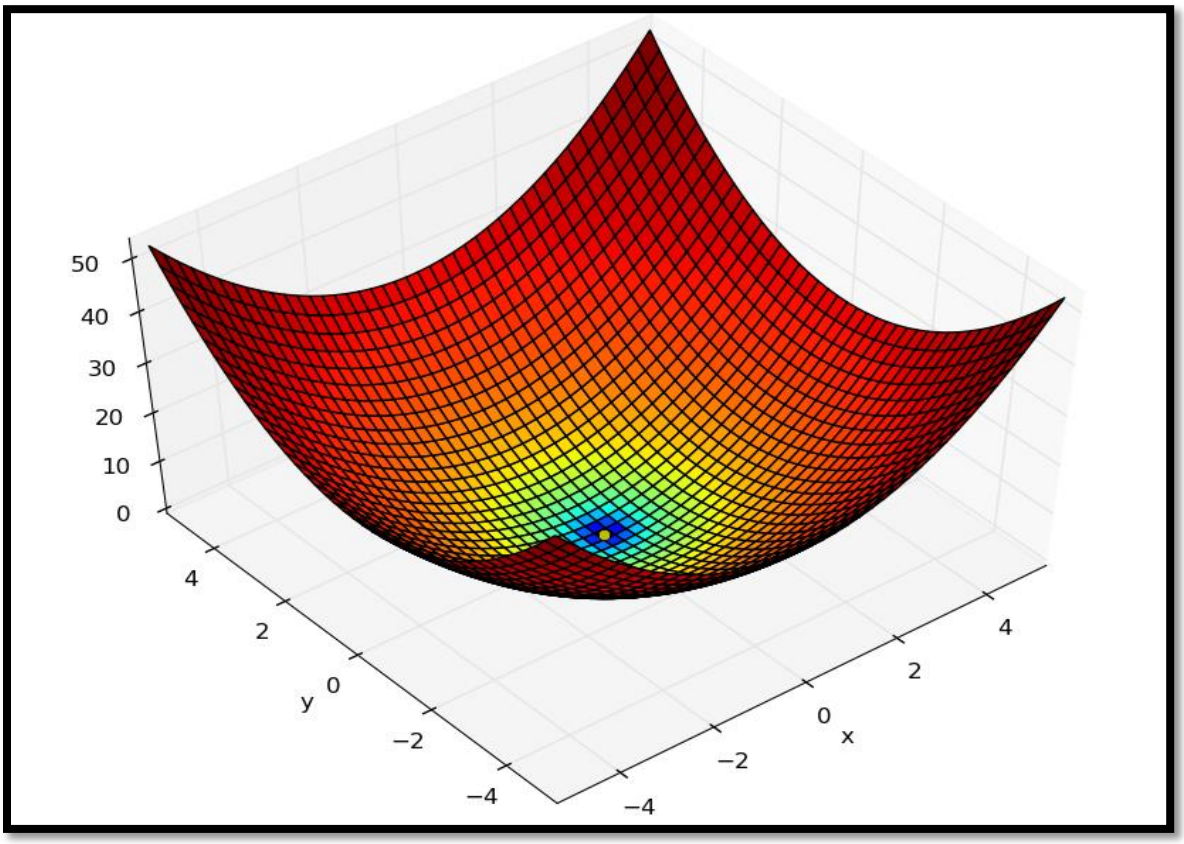

Figure 3.1: Python plot showing the sphere function between -5.12 and 5.12 with the global minimum represented by the yellow dot.

3.2.2 The Rosenbrock function

Rosenbrocks valley is a classic optimization problem, which is also known as the banana function or the second function of De Jong. It has a global optimum located inside a long, flat, narrow, parabolic shaped valley. It is trivial to find the valley itself; however convergence to the global optimum is difficult which is why this problem has been frequently used to test the performance of optimization algorithms. It has the following definition in two dimensions:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

The test area for this function is generally restricted to the region -2.048 ≤ x,y ≤ 2.048, and has a global minimum of $f(x) = 0$ located at (x,y) = 1 see Figure 3.2 (Molga and Smutnicki, 2005).

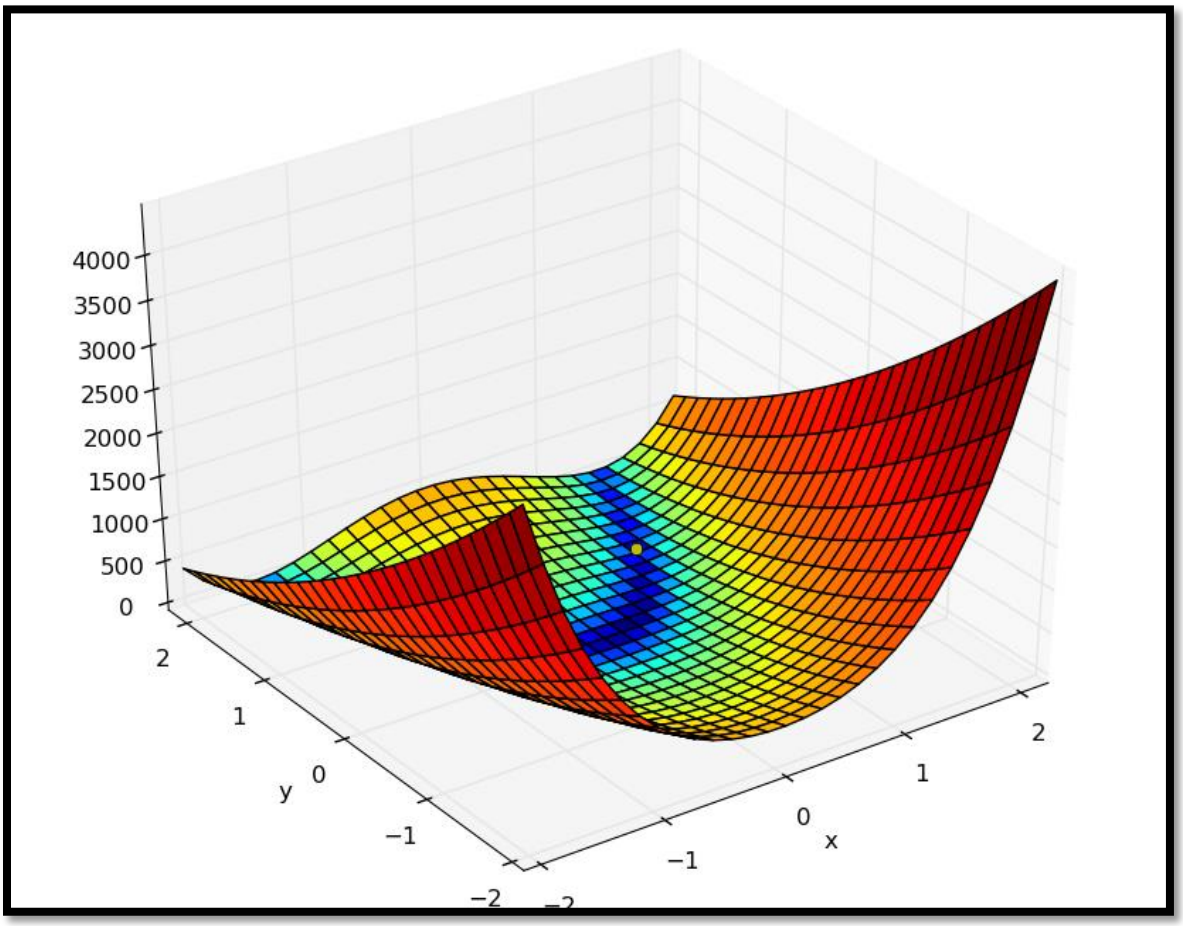

Figure 3.2: Python plot showing the Rosenbrock function between -2.048 and 2.048 with the global minimum represented by the yellow dot.

3.2.3 Rastrigins function

Rastrigins function is based on the sphere function of De Jong with the addition of cosine modulation in order to produce frequent local minima. This makes this test function highly multimodal and the locations of the minima are regularly distributed. It has the following definition in two dimensions:

$$f(x,y) = 10 * 2 + [x^2 - 10\cos(2\pi x)] + [y^2 - 10\cos(2\pi y)].$$

The test area for this function is generally restricted to the region -5.12 ≤ x,y ≤ 5.12, and has a global minimum of $f(x) = 0$ located at (x,y) = 0.0 see Figure 3.3 (Molga and Smutnicki, 2005).

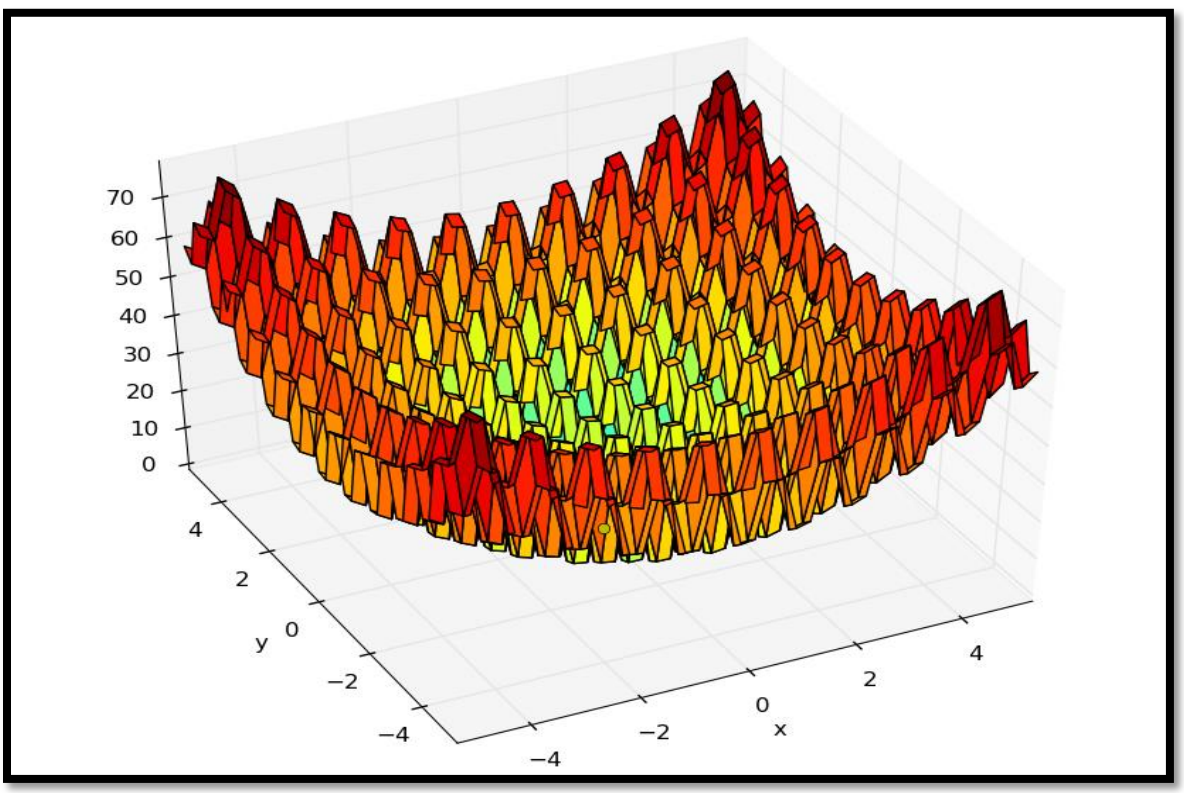

Figure 3.3: Python plot showing Rastrigins function between -5.12 and 5.12 with the global minimum represented by the yellow dot.

3.2.4 The Griewangk function

The Griewangk function is similar to the function of Rastrigin since it is also based on the sphere function with the addition of some cosine complexity. It has many widespread local minima regularly distributed similar to Rastrigins function. It has the following definition in two dimensions:

$$f(x,y) = \frac{(x^2+y^2)}{4000.0} - \cos(x)\cos(\frac{y}{\sqrt{2}}) + 1.$$

The test area for this function is generally restricted to the region -600 ≤ x,y ≤ 600, and has a global minimum of $f(x) = 0$ located at (x,y) = 0.0. The function interpretation changes with the scale; the general overview suggests a convex function similar to the sphere function, a medium-scale view suggests the existence of local minima, and finally zoom on the details indicates complex structure of numerous local minima. This change from smooth convex function to complex structured function can be seen in Figures 3.4,and 3.5 as you change the scale from large, -600 ≤ x,y ≤ 600, to small -5.12 ≤ x,y ≤ 5.12 (Molga and Smutnicki, 2005).

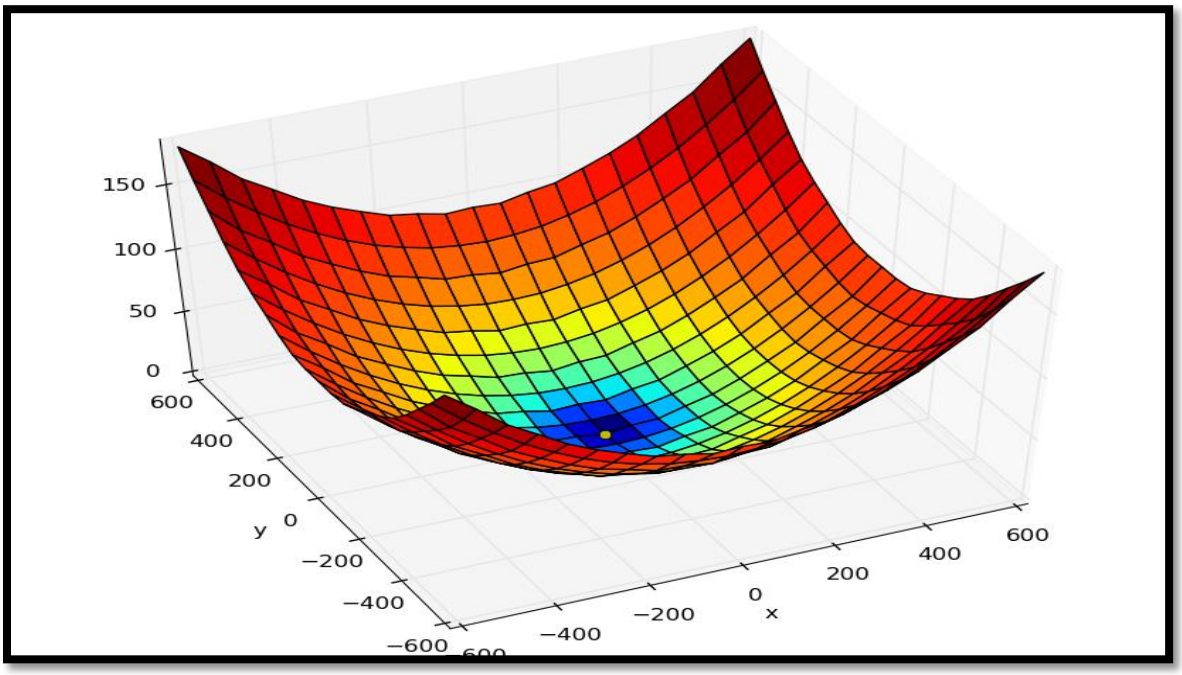

Figure 3.4: Python plot showing the Griewangk function between -600 and 600 with the global minimum represented by the yellow dot.

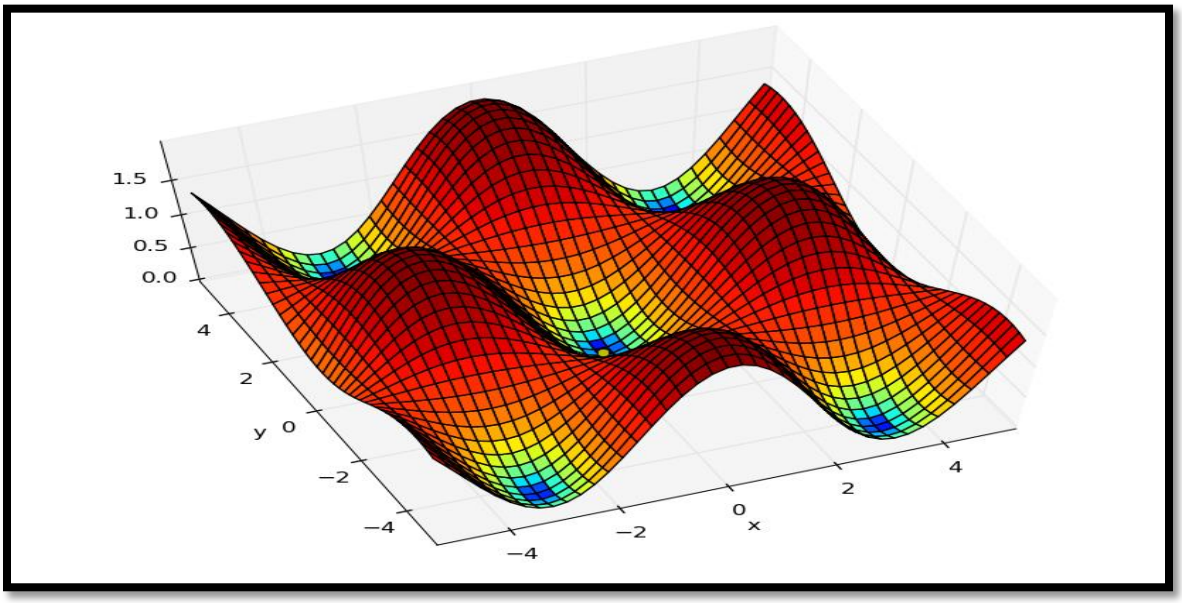

Figure 3.5: Python plot showing the Griewangk function between -5.12 and 5.12 with the global minimum represented by the yellow dot.

3.2.5 The Schaffer F6 Function

The Schaffer F6 function is designed to have a global minimum at the origin surrounded by circular "valleys", as seen in Figures 3.6 and 3.7, that trap local search based methods. This function is considered as a challenging problem for optimization algorithms. It has the following definition in two dimensions:

$$f(x,y) = \frac{0.5+(\sin\sqrt{x^2+y^2})^{\,2} - 0.5}{(1.0+0.001(x^2+y^2))^2}\,.$$

The test area for this function is restricted to the region -600 ≤ x,y ≤ 600, and has a global minimum of $f(x) = 0$ located at (x,y) = 0.0 (Battiti et al., 2005).

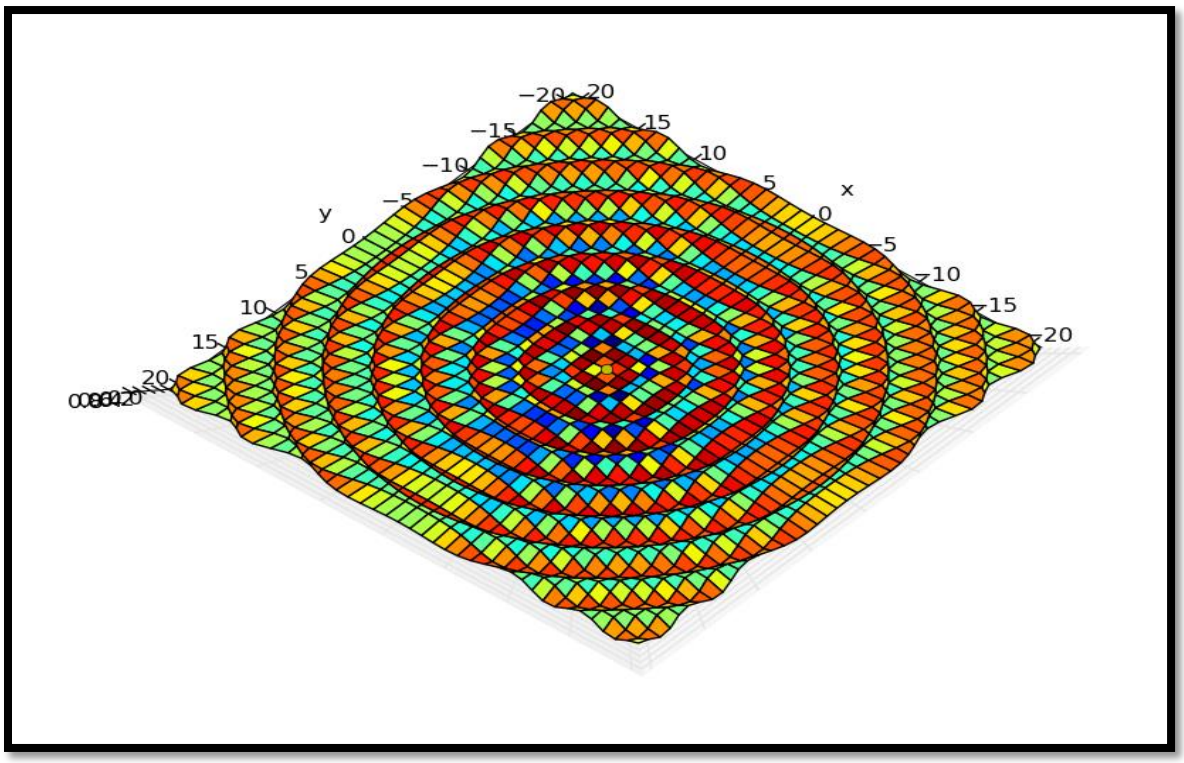

Figure 3.6: Python plot showing the top view of the Schaffer F6 function between -20 and 20 with the global minimum represented by the yellow dot.

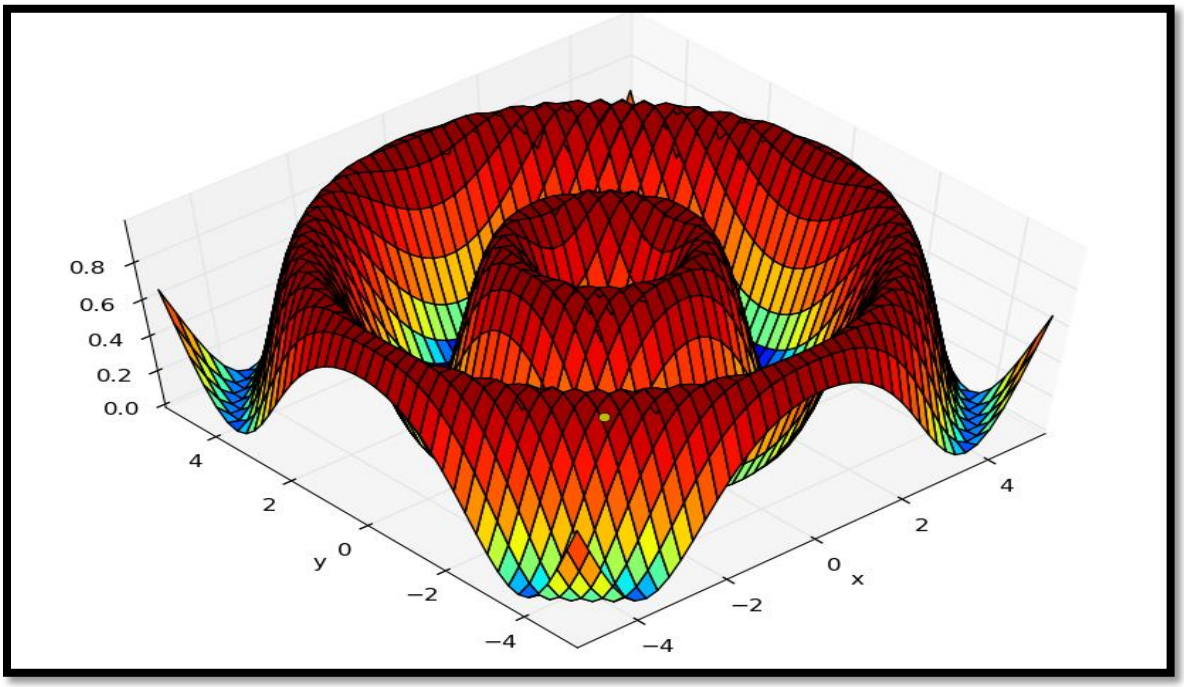

Figure 3.7: Python plot showing the Schaffer F6 function between -5.12 and 5.12 with the global minimum represented by the yellow dot.

3.3 Minimization results and comparison

The results obtained through minimization of these benchmark functions were very promising. All the functions were minimized in two dimensions except for the sphere function, which was minimized in both two and thirty dimensions. This was done in order to create comparable results to those found in Carlisle and Dozier (2001), who used thirty dimensions for all but the Schaffer F6 function.

3.3.1 Minimization results for the sphere function.

In two dimensions the results are quite good. The code converged twenty out of twenty times, a 100% success rate. The average number of iterations it took to find a solution, within 1*e-3* error, was 33.0 and to find the absolute minimum (to within machine accuracy) the average number of iterations was 9130.9. It is clear that finding a solution within acceptable error was easy but finding the absolute minimum was quite a bit harder for this function as can be seen in Figure 3.8, a plot showing an example of the 2D Sphere and the path of the *gbest* particle location as it converges to the minimum.

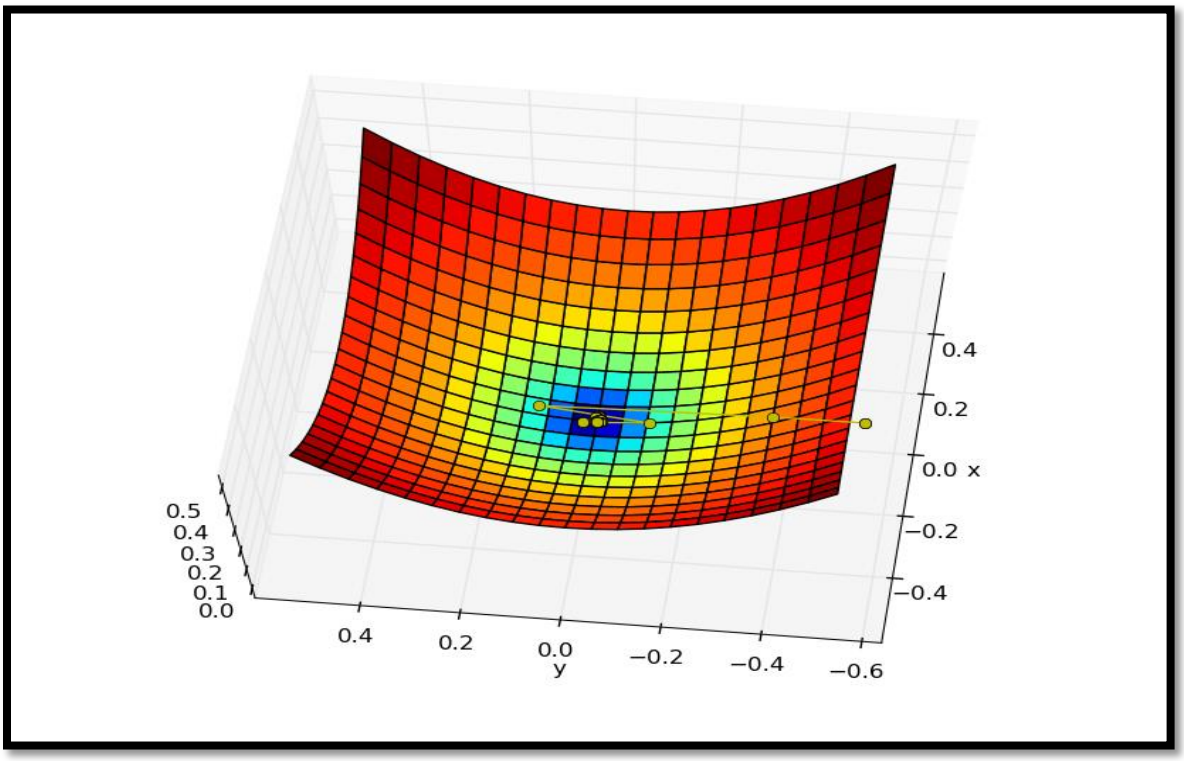

Figure 3.8: Plot of 2D sphere and an example convergence path.

This figure shows a small amount of samples leading into the low region around the minimum, and a dense search around it. For this particular case an acceptable solution was found in 31 iterations, and the absolute minimum was found in 8557 iterations. This fact is also apparent from the results in thirty dimensions, where it never converged to the absolute minimum over the twenty test runs but did converge to acceptable solutions 100% of the time. Using an error of 1*e-2*, as was the case in Carlisle and Dozier (2001), the minimum was found in 666.8 iterations on average. Comparing these results to those of Carlisle and Dozier it is clear that this implementation works well. They record solutions found between 2233.0 and 226.5 average iterations, depending on the set of parameters that they use. The parameters that are most similar to those used here produced results of 361.5 average iterations, which is about two times better, but still very close.

3.3.2 Minimization results for the Rosenbrock function.

The tests done on this function were done in two dimensions with an acceptable error of 1*e-3*. Like the 2D sphere runs for this function also converged successfully 100% of the time. The average number of iterations it took to find an acceptable solution was 60.2, and 846.2 to find the absolute minimum. Similar to the 2D sphere PSO finds the area of the minimum quite fast and then takes some time to find the absolute minimum. This is expected since it is located in a long nearly flat parabolic valley. This can prove difficult to navigate, as can be seen in Figure 3.9, where the convergence path starts towards the minimum (identified as the red triangle) but passes it by, before returning to it several iterations later.

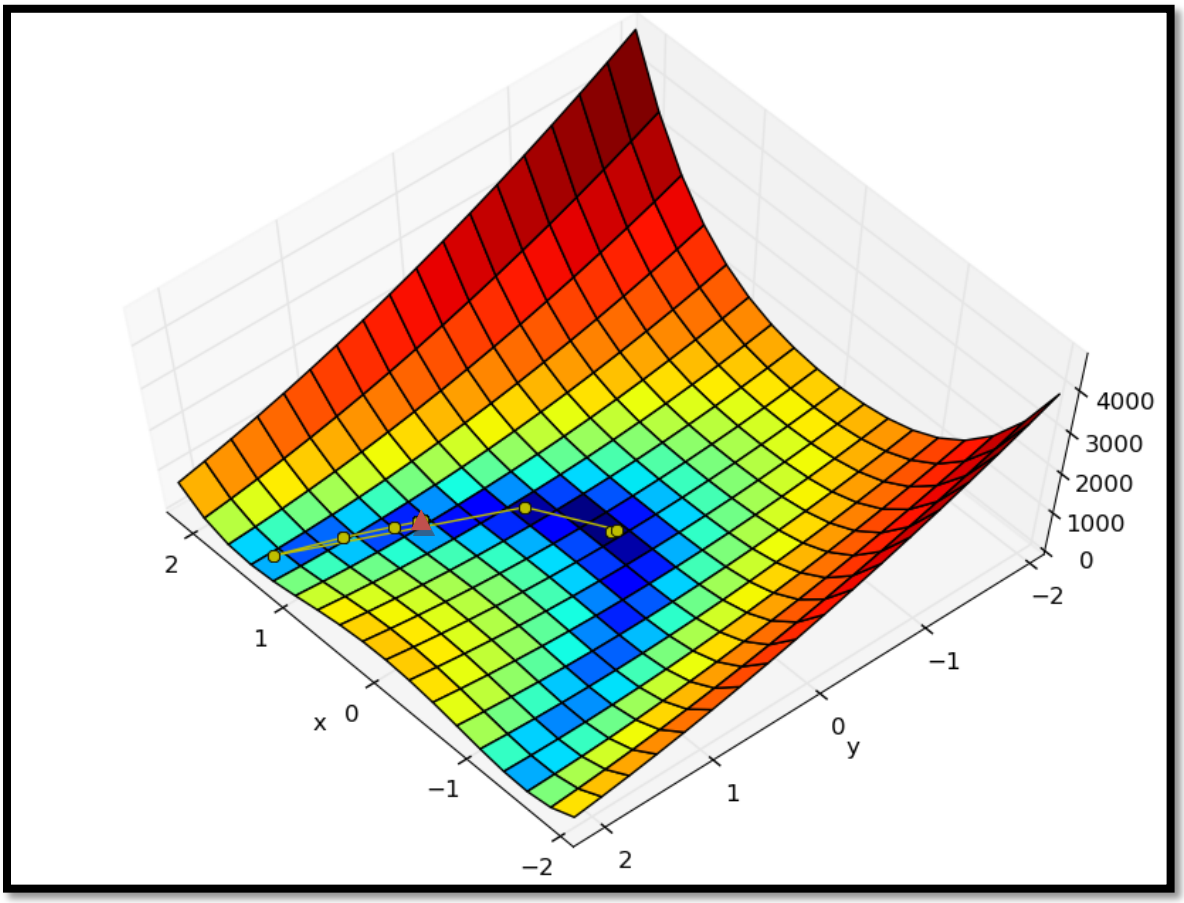

Figure 3.9: Plot of the Rosenbrock function and an example convergence path, where the yellow path is the path of Gbest.

3.3.3 Minimization results for Rastrigins function.

The acceptable error for this function was 1*e-3*. This function is similar to the sphere, except it is littered with local minima. For this reason PSO did not manage to converge for every trial, but did manage to achieve a 90% success rate. The average number of iterations it took to find an acceptable solution was 116.5, and 446.6 to find the absolute

minimum. In this case PSO took slightly longer to find a solution than for the sphere, 33 up to 116 iterations, but there was significant improvement when finding the absolute minimum, 446 down from 9130 iterations. This is most likely caused by the shape of the functions, since it is jumping from one local minimum (i.e. the function is not monotonic) to the next in the Rastrigins function (see Figure 3.10). Whereas in the case of the sphere, PSO is refining one area, which could indicate that this PSO or this set of parameters is better suited for global as opposed to local searching.

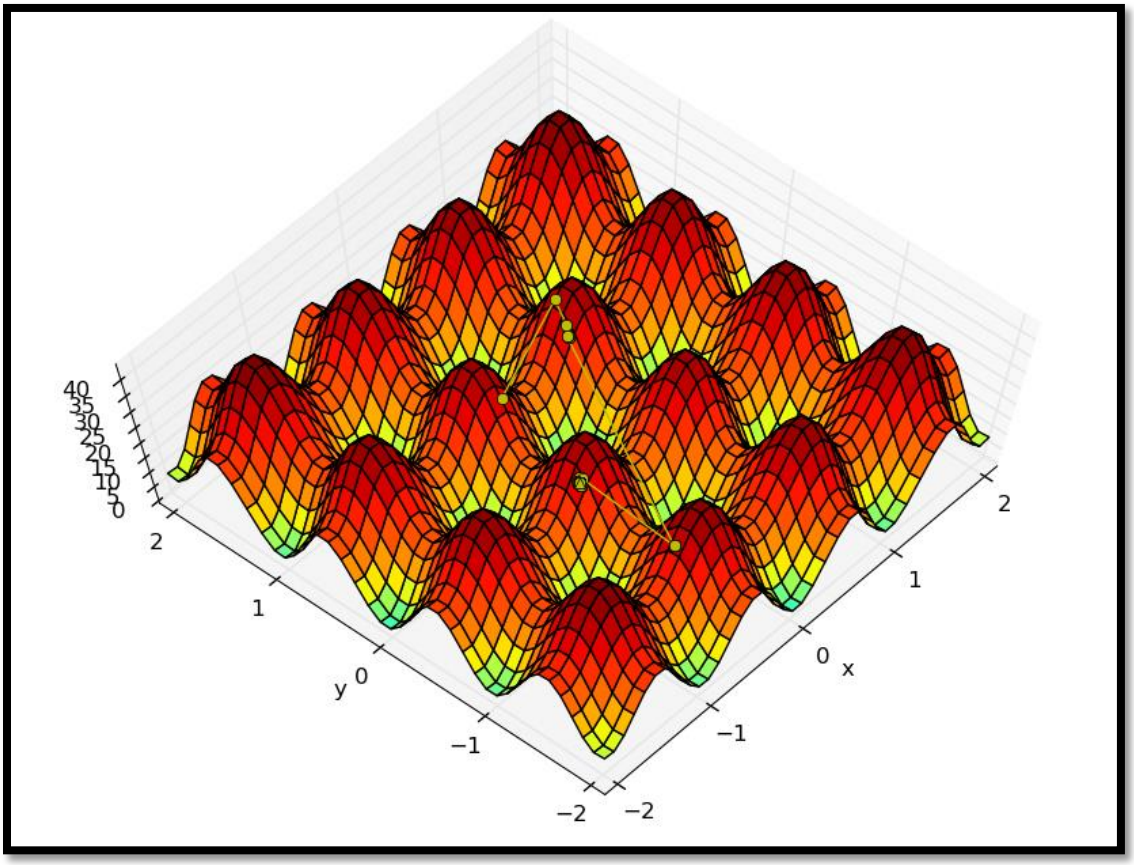

Figure 3.10: Plot of the Rastrigins function and an example convergence path.

3.3.4 Minimization results for the Griewangk function.

Following the Rastrigins function tests, the acceptable error for this function was also 1*e-3*, and it also failed to converge during every trial. The success rate was 80% on this function with an average of 31.1 iterations to find acceptable solutions and 239.2 iterations to find the global minimum. Similar convergence behavior to the Rastrigins function is observed for this function, which is expected since the functions are of similar design. Figure 3.11 shows a convergence path jumping from one local minimum to another, before finally locating the global minimum.

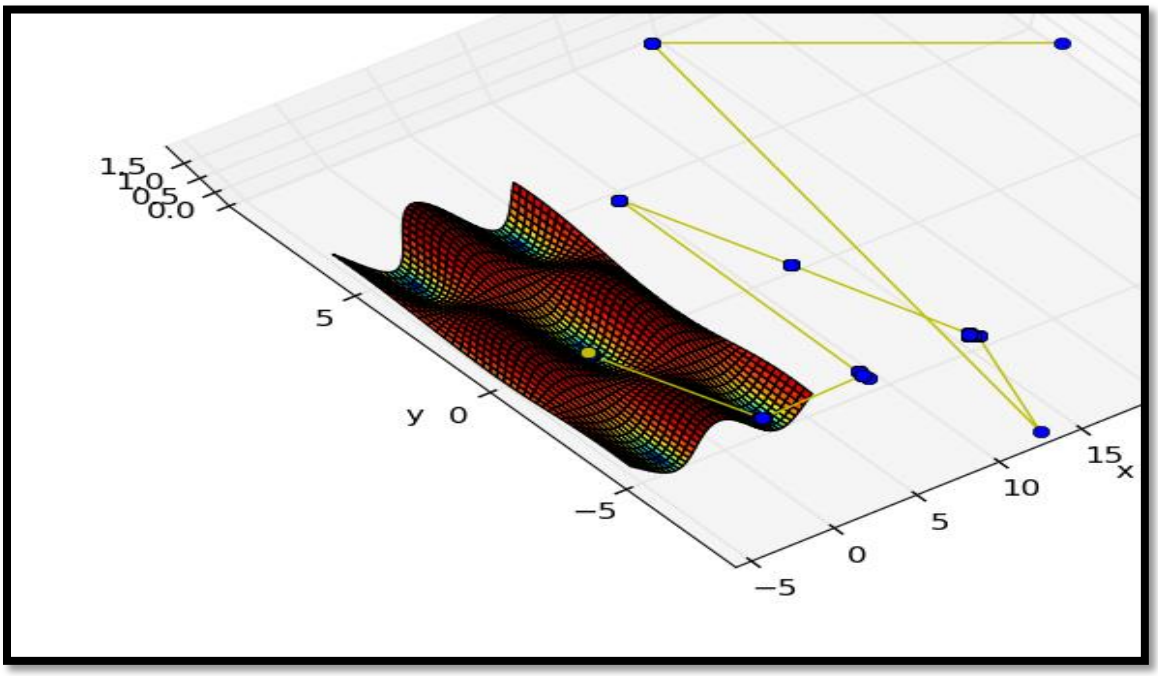

Figure 3.11: Plot of the Griewangk function and an example convergence path. The blank area is un-plotted area of the function but is still part of the function

3.3.5 Minimization results for the Schaffer F6 function.

For the Schaffer F6 function, which is considered to be a difficult test function, the successful rate of convergence was 85 %. The acceptable error used for convergence of this function was 1*e-4* since that is the error used by Carlisle and Dozier (2001). The results they recorded range between a success rate of 100% and an average of 1023 iterations down to a success rate of 85% and an average of 272 iterations. The set of parameters that are most similar to those used here produce results of 85% success and an average of 404 iterations. These are comparable to the results produced during these tests; the success rate was also 85% and the average numbers of iterations were 2255.0 to find acceptable solutions and 2722 to find the global minimum. Figure 3.12 shows the particles jump from one ring to the next, sometimes staying in the same ring, but eventually finding the global minimum in the center ring. This behavior is similar to the previous two examples, both with numerous local minima which are excellent examples of the strength of PSO to avoid being trapped.

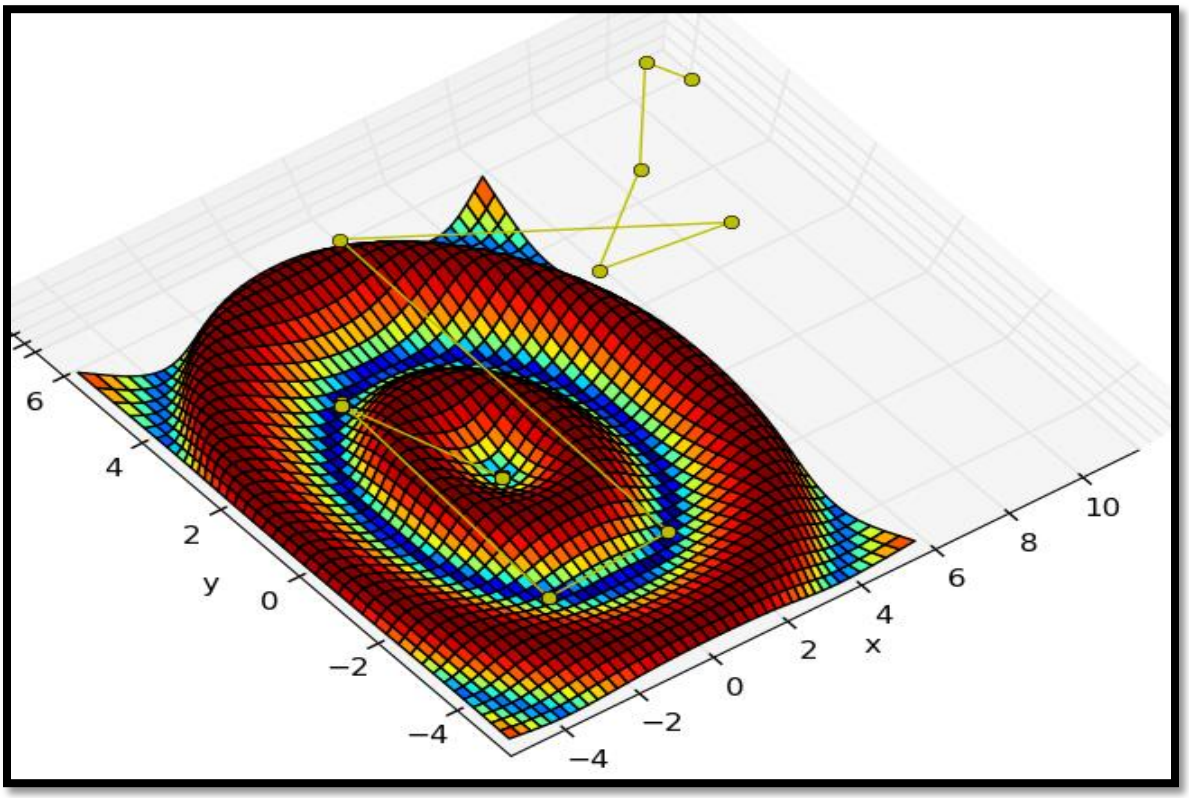

Figure 3.12: Plot of the Schaffer F6 function and an example convergence path.

3.4 Conclusions.

In every case the PSO algorithm tested here performed well, converging 100% of the time in half of the tests and was never lower than 85% convergence in the rest of the tests. The tests in which the results couldn't be compared to other results show that this PSO algorithm performs quite well, it finds acceptable solutions with good success rates and in relatively low number of iterations, but usually takes the most time narrowing down the absolute minimum. This could indicate that it performs global searching well, but has some difficulty with local searching. When compared to the results obtained by Carlisle

and Dozier (2001) these conclusions are further corroborated. In both cases the results were accurate in success rate, achieving 100% for the sphere and 85% for the Schaffer F6 function. The average number of iterations was comparable in the case of the thirty dimensional sphere, where it took about twice as long to find solutions but the Schaffer F6 proved more difficult with around 5 times more iterations than the results of Carlisle and Dozier. This difference in convergence rate is outweighed however by the behavior shown in the three examples with multiple local minima. This avoidance of local minima is exactly the sort of behavior that PSO is used for, and since it is evident in all cases with numerous local minima it can be concluded that this PSO algorithm is working well.

Chapter 4 : Sides Inversion Method

This chapter discusses the conversion of the code from a function minimization algorithm to a geophysical inversion algorithm. The chapter first discusses forward modeling theory, as it will be applied in this thesis, then the differences that were addressed in the conversion process. Model parameterization is a major area of interest in this thesis, thus an introduction to the "sides" method of parameterization is presented in this chapter. Using the sides method and forward modeling, synthetic data was created for the four and five-sided inversion targets. Synthetic data is data that is created or computed, abd is not real world data. Constraint handling methods are also an area investigated in this chapter, and after this the results of the test carried out are detailed and discussed. The source code can be found in Appendix B, along with all results not explicitly shown in this chapter.

4.1 Forward modeling theory

Chapter 1 introduced the forward modeling method, describing it as a three step process. First, make an initial guess based on prior information, then calculate the anomalous gravitational field of the body, and finally adjust the model to better fit the anomaly. A useful way to approximate geologic situations is to replace the cross-sectional shape of two dimensional bodies with simplified polygons. The gravitational attraction of the two dimensional body then depends on the position of the corners of the polygon (Blakely, 1996). If one imagines N lines drawn from an observation point on the surface to each corner of the polygon, the gravitational attraction depends on the length of the lines, $\boldsymbol{r_n}$, and the angle they make with the horizontal, $\boldsymbol{\theta_n}$, as seen in Figure 4.1.

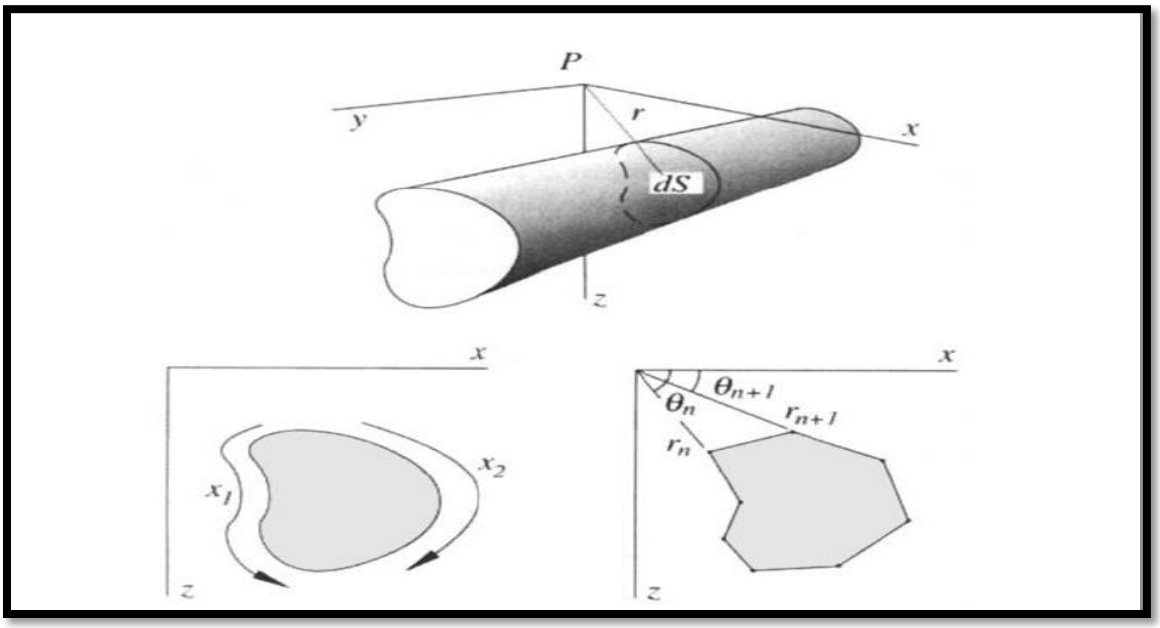

Figure 4.1: Approximation of cross sectional shape using a polygon (Blakely, 1996).

Using these assumptions the gravitational attraction at each observation location can then be calculated using (Blakely, 1996):

$$\mathbf{g = 2^* \gamma^* \rho \sum_n (\beta n / (1 + \alpha n))[\ln(rn + 1 / rn) - \alpha n (\theta n + 1 - \theta n)]} \qquad \mathbf{(4.1)}$$

where γ is Newtons gravitational constant, 6.67x10^{-}11(N), ρ is the density contrast between the body and its surroundings, and alpha and beta are:

$$\boldsymbol{\alpha_n = \left(\frac{x_{(n+1)} - x_n}{z_{(n+1)} - z_n}\right)} \text{ and } \boldsymbol{\beta_n = x_n - \alpha_n \times z_n} \qquad \mathbf{(4.2)}$$

Using this equation will give the gravitational attraction of the body in question at one observation location; this is repeated at each station in the survey to create an anomalous gravity signature.

4.2 Converting function minimization code to an inversion algorithm

Modifying the code used for the special benchmark functions in Chapter 3 into an inversion algorithm requires four key changes: 1) an increase in the number of parameters used, 2) incorporation of a Forward method used to calculate predicted data, 3) calculation of a misfit function, and 4) addition of a penalty function or constraint-handling method.

4.2.1 Inversion targets.

The first step towards creating a PSO inversion algorithm was to increase the number of parameters in the existing code. Previously it used two parameters, which were the x and y coordinates of the particle, only for the sphere example. To evolve into an inversion code, the number of parameters needed would depend on the target sought during an inversion. The initial inversion target was a box-shaped object with a uniform density value (Figure 4.2). This initial object and the region in which it is "buried" were created with simplicity in mind rather than geologic significance. That said, the survey region consisted of a 60m long survey line in the x direction between x = -30m and x = 30m, and 100m in the z direction from the "surface" z = 0m down to z = -100m. Synthetic data were calculated at 0.5m intervals along the suface, giving a data set of 120 readings. The initial target model, a rectangle with dimensions of 20m by 10m, was placed in this region between x = -10 and 10 in the x direction and between z = -20 and -30 in the z direction with a density value of 1000 kg/m^3. Therefore the number of parameters sought in the inversion would be five, one for every corner of the object and one for the density.

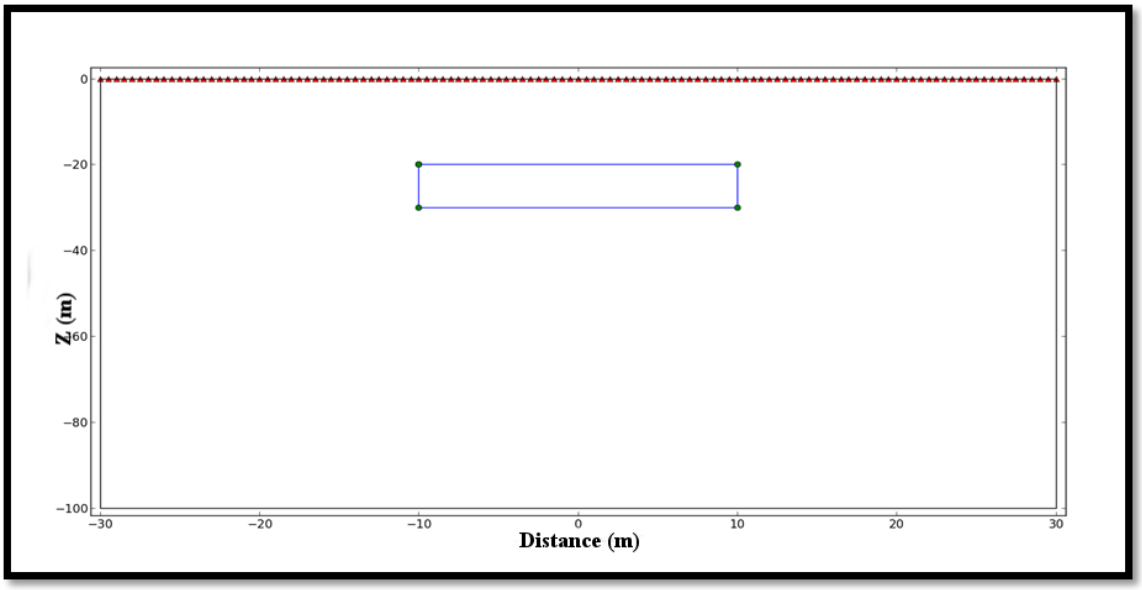

Figure 4.2: Initial inversion target (blue) as well as boundaries of the survey region (black box), with the observation locations (red triangle).

The second inversion target considered in this chapter had an additional corner (Figure 4.3), making it slightly more complex and adding some difficulty to the overall inversion. It was created by adding a peak in the middle of the box to generate a house shape which extends towards the surface. The house has the same density value as the box shape (1000 kg/m^3), and is located in the same area, with its peak extending to the location x = 0.0m, z = -10m.

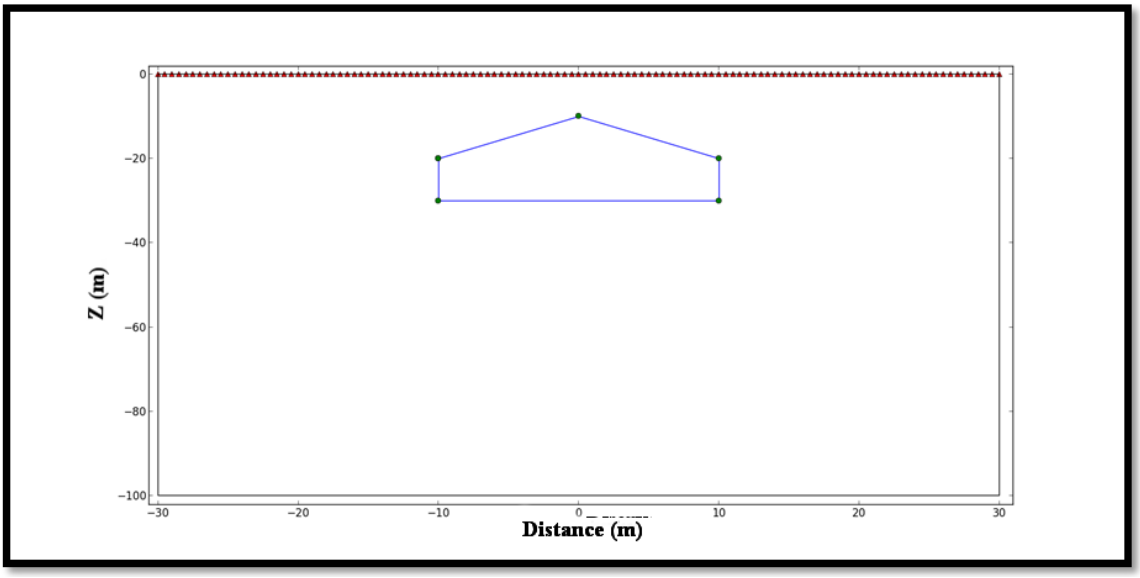

Figure 4.3: Second inversion target considered in this chapter, five sides (blue)

4.2.2 Model parameterization.

As mentioned previously, the inversions carried out during the course of this thesis are non-linear. The parameters being sought during a non-linear inversion are those that specify the shape and density of the polygon. The two methods investigated both define a shape and a density for that shape, but they differ in the manner that they specify the shape. The first method, henceforth called the "sides method", was the first attempt at defining shapes in a way that could be used for inversion. The sides method parameterized the objects using the locations of sides and density, thus five parameters in total, as seen in the sample layout below:

Swarm[i] =[x1,x2,z1,z2,rho]

The corners are found through a combination of an x and a z coordinate. The left and right sides of the object are specified by their x coordinates, and the top and bottom sides

are specified by their z coordinates (seen in Figure 4.4). The corners are then defined by the intersection of the x positions and the z positions in a clockwise rotation as follows: Top left = [x1,z1], Top right = [x2,z1], Bottom right = [x2,z2], Bottom left = [x1,z2]

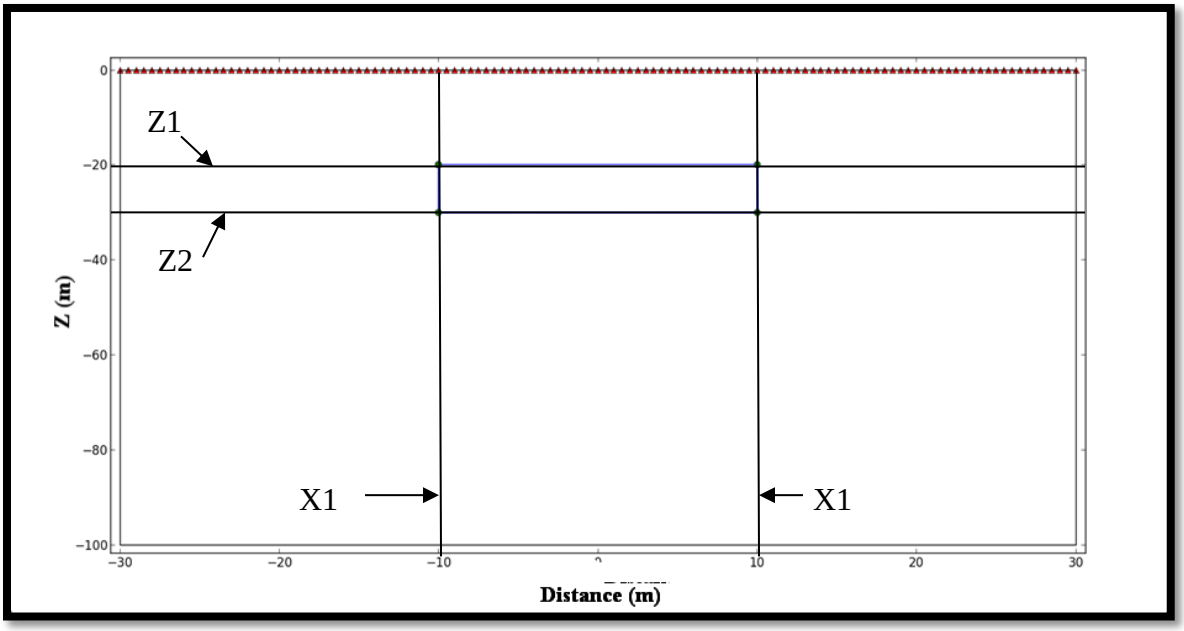

Figure 4.4: Diagram showing how parameterization of four-sided objects, using the sides method, is performed.

Defining the indices of the five-sided objects is done in a similar manner to that for the four-sided objects with two simple differences, there are 2 additional parameters:

Swarm[i] =[x1,x2,x3,z1,z2,z3,rho]

where x3 and z3 are the x and z coordinates of the additional vertex (see Figure 4.5). Since the algorithm works in a clockwise order, the corners become: Top left = [x1,z1], Top Middle = [x2,z2], Top right = [x3,z1], Bottom right = [x3,z3], Bottom left = [x1,z3]

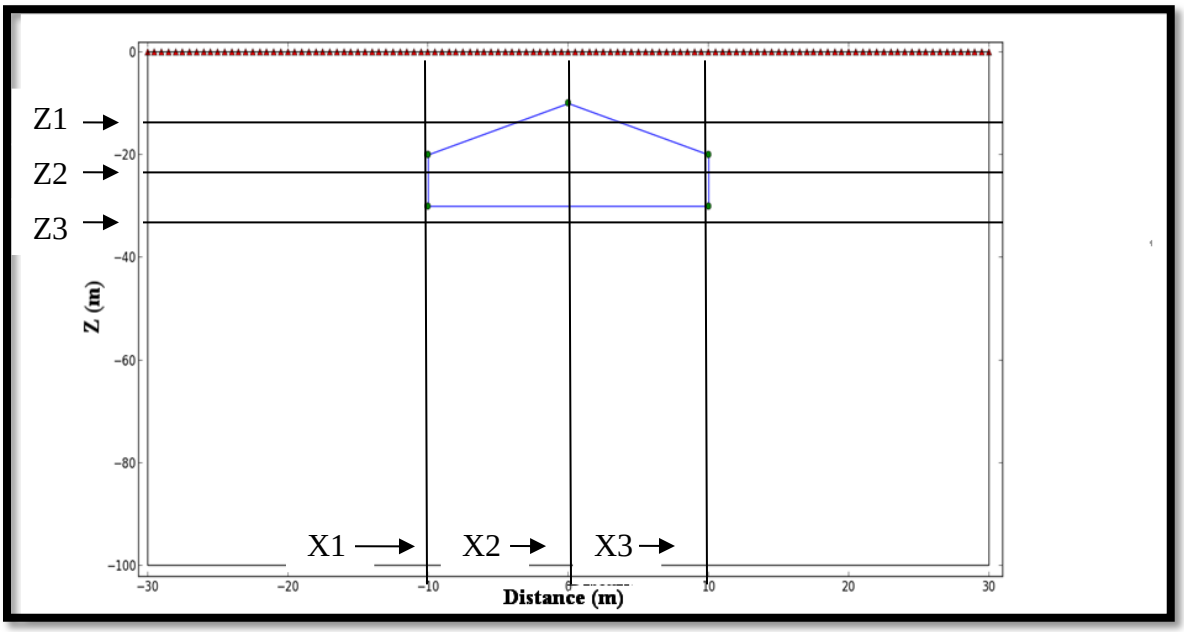

Figure 4.5: Diagram showing how parameterization of five-sided objects, using the sides method, is performed.

The parameterization was done in the way described above for two reasons: it was the simplest transition from the previous function implementation, and it provides a constraint on the possible shapes taken by the anomalous region during an inversion. The anomalous region is constrained due to the shared coordinates of the sides. These shared sides, and the inherent constraint they applied, was one of several reasons for the creation of a new model parameterization technique, which will be covered in the next chapter. The length of the swarm as well as the *pbest* and *gbest* best arrays simply needed to increase in length from two to five parameters, which would allow the use of shapes in the minimization process.

4.2.3 Synthetic data and data misfit function.

Once the algorithm can operate with an increased number of parameters, forward modeling and data misfit calculations replace the benchmark function. Using the forward method discussed in Section 4.1, synthetic data for a particular model were computed. The profiles of synthetic data for both the four and five-sided targets are bell shaped curves, but are not quite the same shape see Figures 4.6 and 4.7 respectively. The data for the four-sided object is a more broad, smooth feature since the object is a consistent depth from the surface, whereas the five-sided object extends towards the surface in the middle creating a more pronounced peak in the profile above that area.

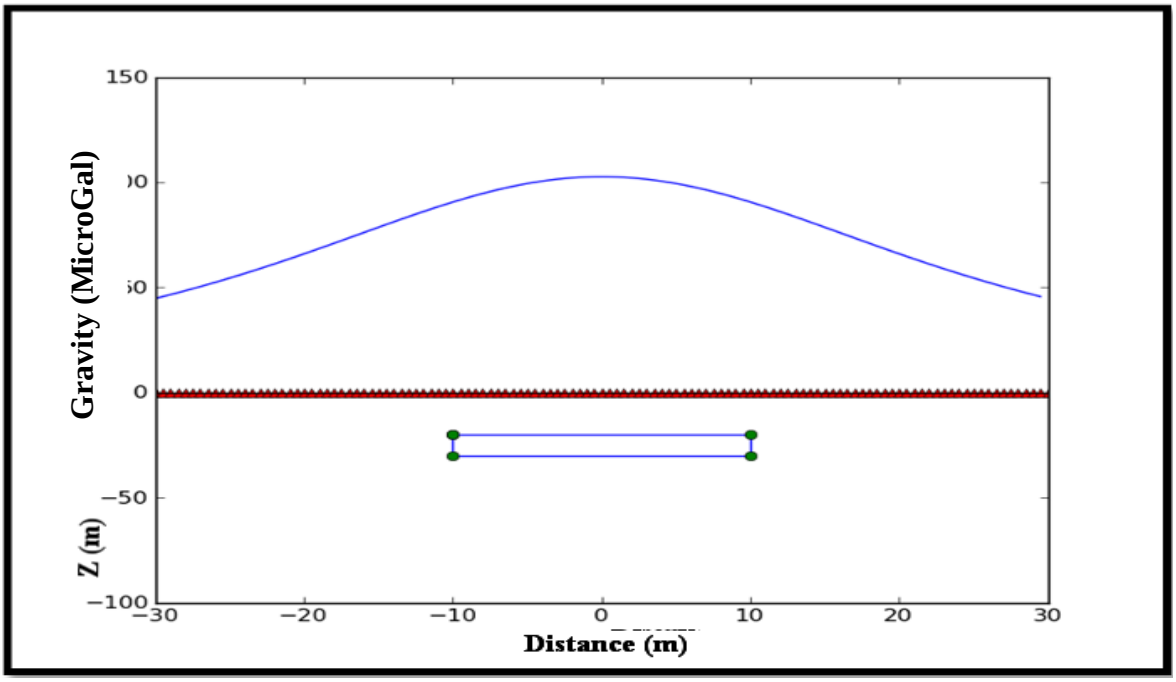

Figure 4.6: Diagram of four-sided model with its synthetic data profile.

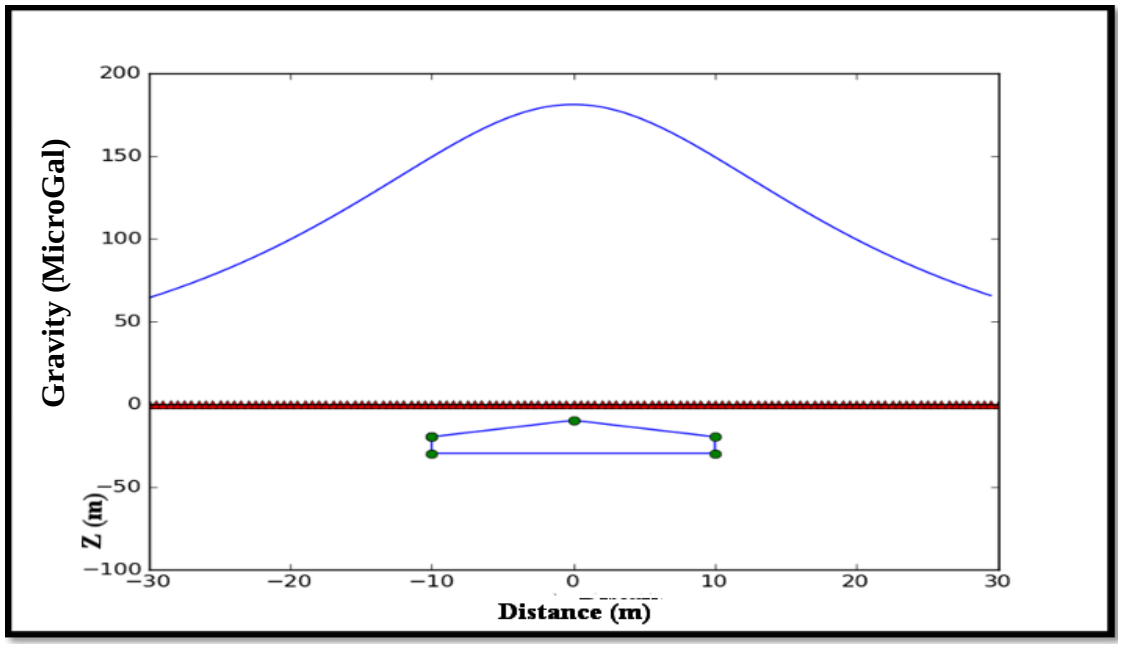

Figure 4.7: Diagram of five-sided model and its synthetic data.

To compare the predicted data to observed data, a misfit function is used. The type of misfit used here is a least squares misfit, which takes the square of the difference between the observed data and the calculated data at each observation location and then sums them together to give one value of misfit. The misfit function is defined as:

$$\boldsymbol{Misfit} = \sum_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{d_i^{obs}} - \mathbf{d_i^{calc}})^{\mathbf{2}}$$

where N is the total number of observations, d_i^{obs} is the ith observed datum, and d_i^{calc} is the ith predicted datum.

4.2.4 Constraint handling methods

Constraints can be either hard or soft in nature, representing physical limits of systems or preferred operating ranges (Coath and Hamalgamuge, 2003). Evolutionary algorithms (EAs) are algorithms which use mechanisms inspired by biological evolution such as reproduction, mutation, recombination, and selection (this includes PSO). When using EAs problems can to be solved as an unconstrained optimization task by effectively removing the constraints and adding things such as penalty functions. According to Coath and Hamalgamuge (2003), these types of constraint handling methods used in EAs fall into several distinct categories, some of which include: 1) penalty functions 2) rejection of infeasible solutions 3) repair algorithms 4) specialized operators and 5) behaviour memory. The categories that were focused on here, and are discussed further in Chapter 5, were the penalty function and rejection of infeasible solution methods, since these have shown promise when implemented with PSO previously. In the early versions of the algorithm implemented here, there was no method in place to constrain the shapes of the models that the algorithm produced, besides boundary issues, which inevitably lead to undesirable outcomes, which will be discussed subsequently. The boundary issues were dealt with via a built in NumPy function called Clip which I incorporated into the algorithms code and which would not allow the values of the swarm to go outside a set range. If a particle were to venture out of the search region it would be stopped using the Clip function and then its direction would be reversed by applying the calculated velocity in the opposite direction, sending it back into the acceptable range for the examples presented in this chapter. The swarm was confined to the search region described in

Section 4.2.1.The effectiveness of this lack of constraints method will be discussed in the results section of this chapter; however there was a problem with the corners rearranging in the four-sided tests and models being turned inside out in the five-sided models. This led to the investigation of methods for preventing these unacceptable results.

The result of this investigation was the “corner switch” method of constraining the shape of the object during inversion. This method is a stationary type of constraint, that is, its conditions never change, in which the X and Z values are monitored as the inversion is running, and instances where the shape has turned itself inside out are caught, the conditions do not evolve with each iteration (see Figure 4.8). An example of being turned inside out would be if the left side value of X was greater than the right side value of X, or if the top Z value is more negative than the bottom Z value. In the case of the X values, it would still appear as a box shape, but the forward method would have to work in reverse. In both cases, the irregular values would be interchanged, restoring an acceptable shape to the object.

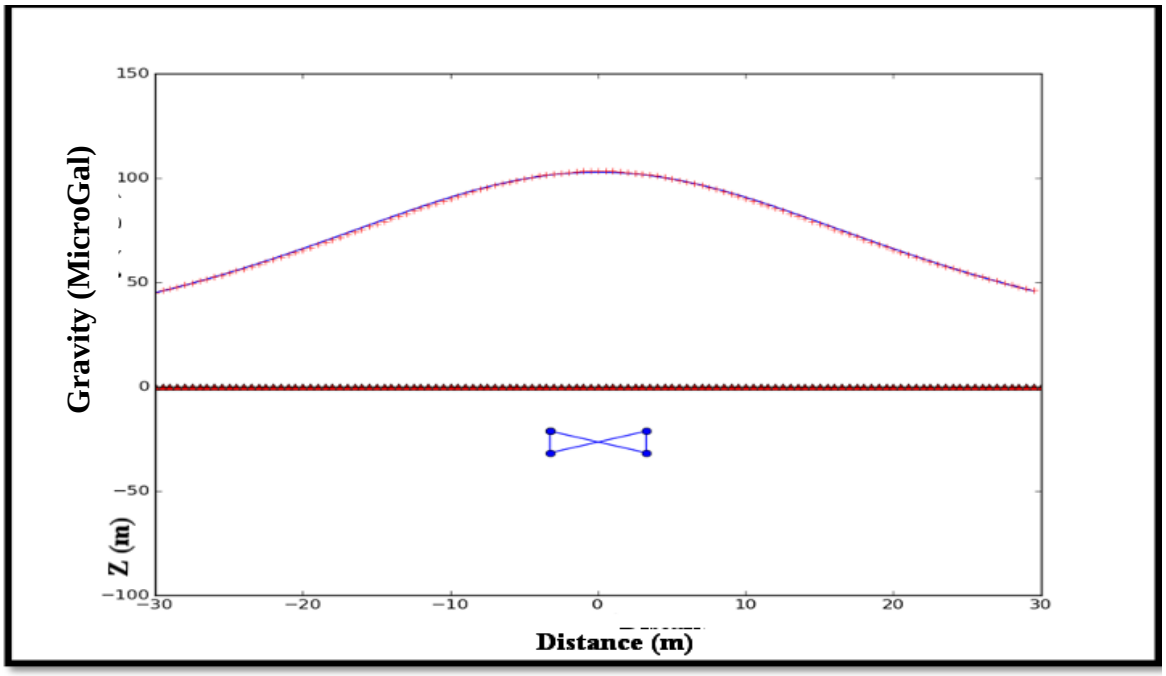

Figure 4.8: Depiction of a four-sided model which has become infeasible by turning inside out, and the predicted data (red xs) it would generate plotted with the synthetic data (blue line).

4.3 Results

This section discusses the results obtained through the tests performed with the non-constrained algorithm on the four and five-sided models, as well as the tests using the corner switching method of constraints on the four and five-sided models. Each set of tests was performed over five thousand iterations, and repeated twenty times. The recorded values were the misfit, the time taken, the X and Z values and the density. The data sets had no noise added, and had 120 data points in total. The success or failure of the inversions will be based on 3 criteria; the shape of the model, the misfit, and the shape of the gravity curve. Any model that is not turned inside out will be considered

acceptable, the lower the misfit values the better, and the shapes of the gravity curves will be compared to the original synthetic curves. Therefore an acceptable inversion will be one that is not turned inside out, has a low misfit value, and generally reproduces the shape of the gravity curve.

4.3.1 Unconstrained four-sided inversions

The results obtained by the algorithm without constraints on the four-sided model were mixed but promising. Because of the non-uniqueness of gravity inversion the number of models that can reproduce a near perfect match to the synthetic data curve, i.e., a misfit close to zero, is infinite. This perfect reproduction did not occur for any of the twenty test inversions; however the results collected show that more than sufficient models can be achieved with misfit values slightly greater than zero. Over the twenty test inversions the average misfit achieved was 41.4. The highest misfit returned was 226, whereas the lowest value returned was 0.1. The misfit value of 0.1 is actually a built in stop parameter which indicates that the inversion has found a model that is unlikely to improve, and no more iterations are needed. The average time taken for an inversion was 50.3 minutes; however, in four out of the twenty inversions the stop parameter was triggered. In these cases the average time taken was only 5.4 minutes and the number of iterations needed averaged 501.

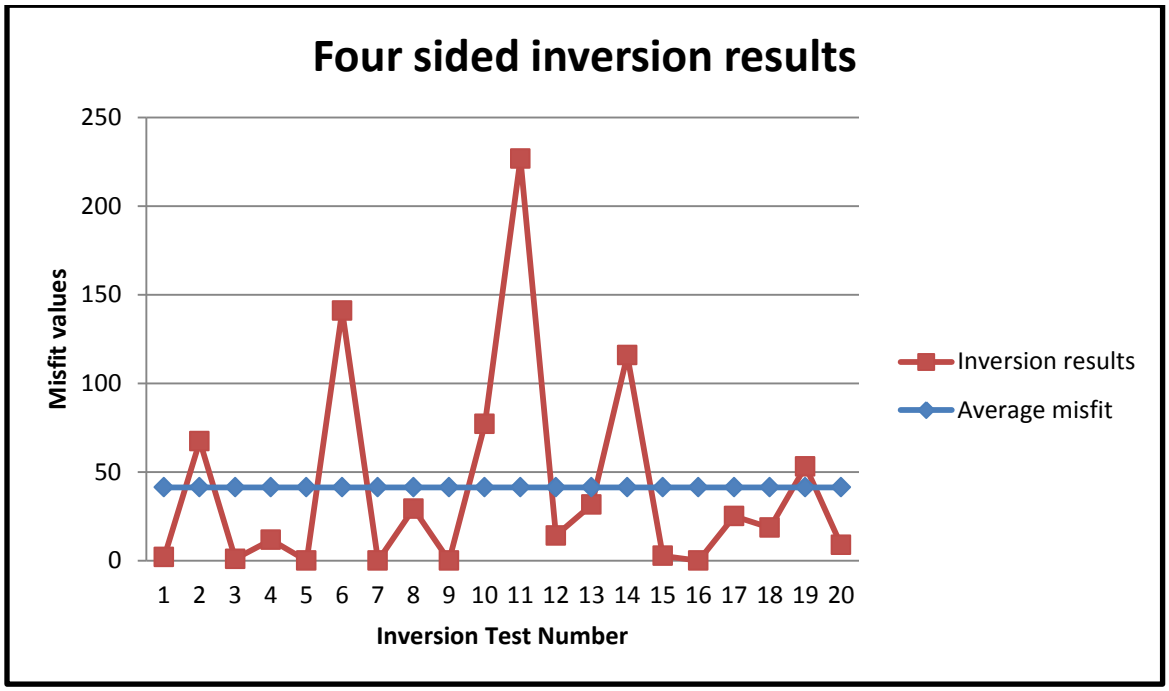

Figure 4.9: Line plot of inversion results returned from tests on four-sided model without constraints.

Figure 4.9 shows the misfit values represented by the red line of every test inversion carried out. The overall average misfit value is indicated by the blue line. The majority of the results lie below the average line; some are near but above the line and there are a few higher outliers, which cause the average to increase. Even though these outliers are higher than the majority of the results, as demonstrated by the following discussion, they are still very acceptable results. Figure 4.10 shows the original model with green corners and the synthetic data as the blue curve. The blue cornered shape is the inversion result with misfit 226 (the "worst" misfit achieved), which creates the gravity curve represented by the red "+" marks. The shape is much narrower in the x direction than the true shape but larger in the z direction than the original. The density value is over

twice that of the original model to compensate for its smaller overall size. The main thing to note is that even though the shapes are quite different there is a large agreement between the gravity curves. The peaks have nearly identical maximum amplitudes, and the shapes of the curves are very similar. The red curve is slightly off center: it can be seen that it is shifted ever so slightly to the right when compared to the blue curve. The cause of this shift is the object also being slightly off center: it extends from x = -1.42m to x = 2.5m, meaning the center of the body is located at x = 0.54m. The predicted curve isn't as good as it could be, since the misfit is 226, but overall it is very close to the synthetic curve in every aspect, implying that this is a good result, and a successful inversion despite being the poorest result of the 20 inversions runs.

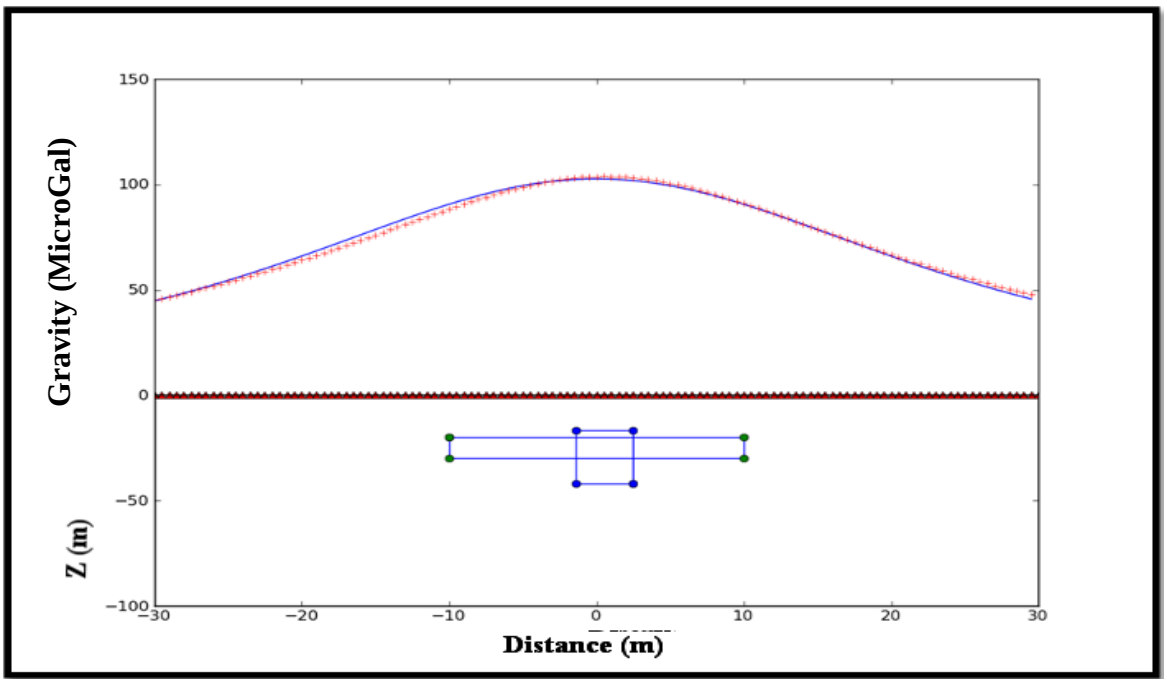

Figure 4.10: Diagram of predicted model from inversion result with misfit 226, with its predicted data (red+), shown with the target model and the synthetic data (blue).

The inversion result that is closest to the average misfit value is shown in Figure 4.11. This model is similar to the model returned with a misfit of 226; although it is wider in the x direction, extending from x = -2.29m to x = 2.19m but does not reach as deeply. The top depth is nearly identical to the depth of the original model but it extends deeper than the original model, reaching a depth of 35.5m. Once again the constructed model has compensated size with an increased density value, which is 2999.9 kg/m^3 for this predicted model. The predicted data however are extremely close to the synthetic data, as can be seen when comparing the red + and blue curve in Figure 4.11. The peak values are identical, and the shapes of the profiles are nearly identical as well.

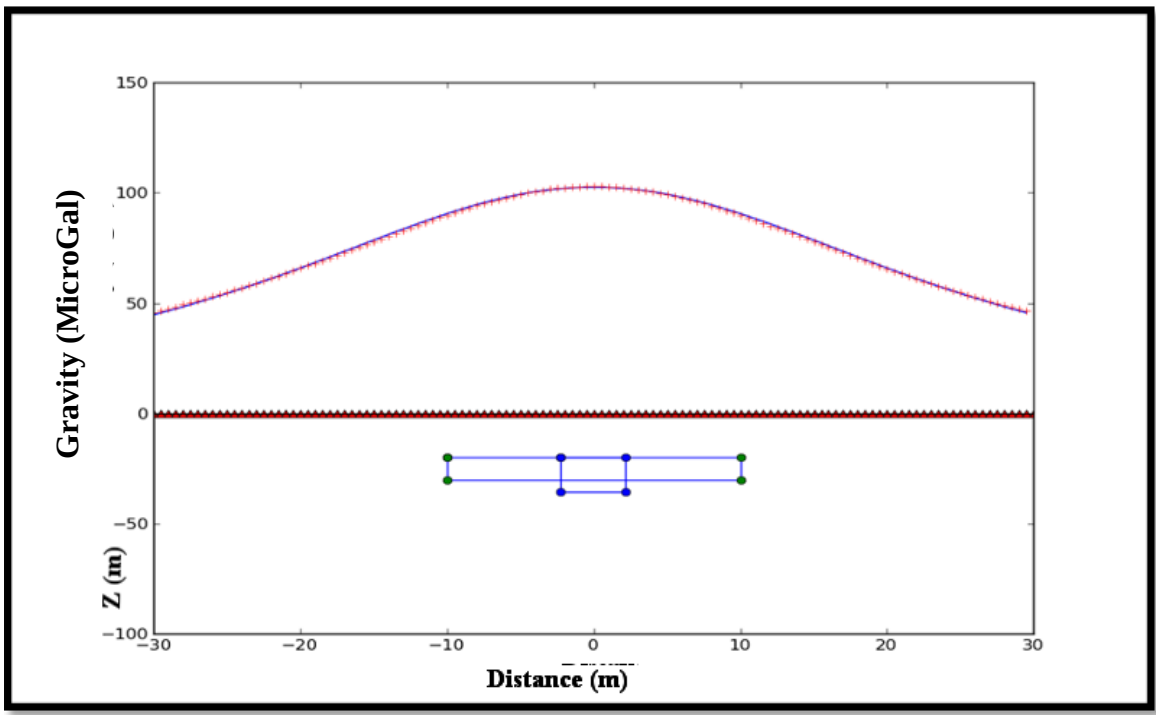

Figure 4.11: Diagram of predicted model from inversion result with misfit closest to average value, with its predicted data (red+), shown with the target model and the synthetic data (blue).

The model which produced the best inversion result is seen in Figure 4.12. Unlike the previous results which over estimated the extent in the Z direction, creating a narrow deep-reaching shape, this result is more similar to the original model. The anomalous region is located between x = -9.49m and x = 9.49m, which is perfectly symmetric about the origin. It has a density of 1885.34 kg/m^3 which compensates for the difference in size of the models and allows the predicted curve to be a near perfect match to the synthetic data curve in every detail even though the misfit isn't quite zero. There were actually four results with a misfit of 0.1 as mentioned previously, all of which are very similar in shape to the constructed shape in Figure 4.12. see comment

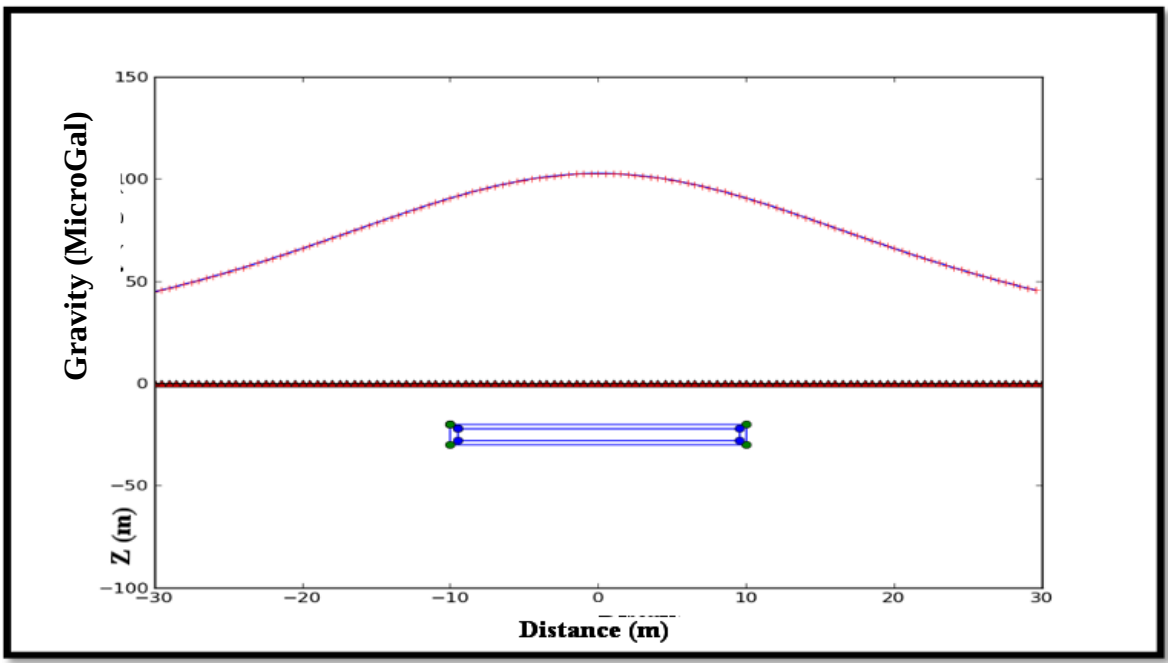

Figure 4.12: Diagram of predicted model from inversion result with lowest misfit value of this set, and its predicted data (red +), shown with the target model and the synthetic data (blue).

Figure 4.13 is a plot of all twenty shapes returned from the inversion tests. The original model is also seen in this figure with the green corners and blues lines as before. The most notable thing about Figure 4.13 is that there are basically two general shapes. The first shape is narrower than the original model but extends deeper, and the second general shape is roughly the same size as the original model in the x direction but doesn't have the same thickness. One thing that all the resulting models have in common is that they have higher densities than the original model, the lowest density being 1484.0 kg/m^3 with an average density of 2305.8 kg/m^3. This increase in density is to compensate for the smaller size of the constructed models.

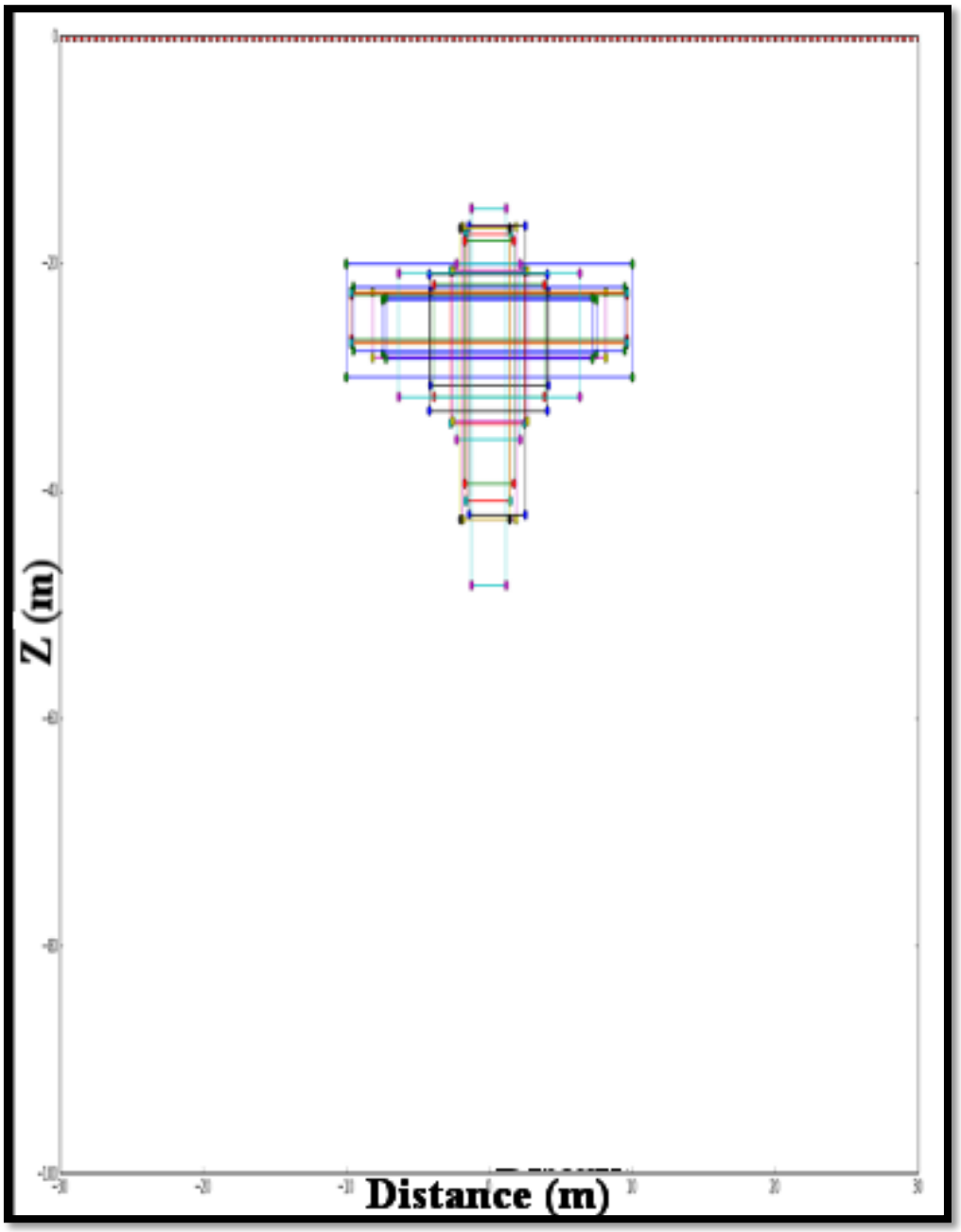

Figure 4.13: Diagram of all model shapes returned for unconstrained four sided inversions.

Another noteworthy feature of Figure 4.13 is the relationship between the shape of the model and the misfit value. Even though these would all be considered successful inversions, some reproduce the original models better than others, and the best results are obtained from models which are more similar to the original model. In Table 4.1 the results highlighted in yellow represent the best results obtained, the ones in blue are very close but have slightly higher misfit, and those in green are the four with the highest misfit. When these results are plotted the relationship between the model shape and misfit value becomes apparent. Figure 4.14 shows the original model, with the green corners and the blue lines as before, and the four best inversion models. The inversion results have a very similar shape to the original model, and are very closely grouped. Figure 4.15 shows the same models as Figure 4.14 with the four next best inversion models added. It can be seen that these models are contracting towards the origin symmetrically, and are generally reaching deeper as they contract. Further evidence of this trend is seen in Figure 4.16 where the best models are replaced by the highest misfit models. The high misfit models generally reach the deepest and are the narrowest. So the trend for the four sided models is being driven by the parameterization and it can therefore be inferred that, at least for the four-sided model, the more narrow and deeply reaching the model, the higher the misfit will be.

Table 4.1: Inversion results of tests performed on four-sided models without constraints. The columns are as follows: Test # is the number order in which the tests were performed, misfit is the misfit achieved by the inversion, X1 is the left side coordinate, X2 is the right side coordinate, Z1 is the coordinate of the top and Z2 is the coordinate of the bottom.

Test #	Misfit	Time Taken(min)	X1	X2	Z1	Z2	Rho	convergence iteration
1	2.17	102.49	7.47	-7.46	27.9	23.17	2855.97	
2	67.48	58.19	1.48	-1.64	40.89	17.39	2914.92	
3	1.1	78.09	8.17	-8.17	28.23	22.5	2147.82	
4	11.84	57.83	4.12	-4.12	30.73	22.44	2997.9	
5	0.1	6.07	9.65	-9.65	26.66	22.81	2675.79	529
6	141	58.7	-1.21	1.18	15.12	48.23	2798	
7	0.1	5.2	-9.65	9.66	22.5	27.05	2267.61	450
8	29.4	50.17	-2.68	2.46	20.61	34.08	3000	
9	0.1	4.7	-9.49	9.49	22.09	27.67	1885.34	495

10	77.15	53.88	-1.84	1.87	16.77	42.49	2248.86	
11	226.7	54.07	2.5	-1.42	42.1	16.63	2139.43	
12	14.23	58.34	3.88	-3.86	31.66	21.87	2714.83	
13	31.65	55.6	-2.29	2.19	19.97	35.51	2999.99	
14	115.9	71.31	-2.01	1.45	16.83	42.43	2430.54	
15	2.77	71.5	7.22	-7.21	28.39	22.92	2564	
16	0.1	6	-9.66	9.66	22.61	26.92	2394.63	531
17	25.14	55.37	-2.55	2.69	20.65	33.88	2982.36	
18	18.7	53.64	-4.2	4.08	20.91	32.95	2067.82	
19	53.25	53.55	1.72	-1.7	39.37	17.98	2895.6	
20	8.9	52.33	-6.32	6.37	20.86	31.69	1484.07	
Average	41.389	50.3515	-------	---------	---------	---------	2523.274	

Table 4.1 continued.

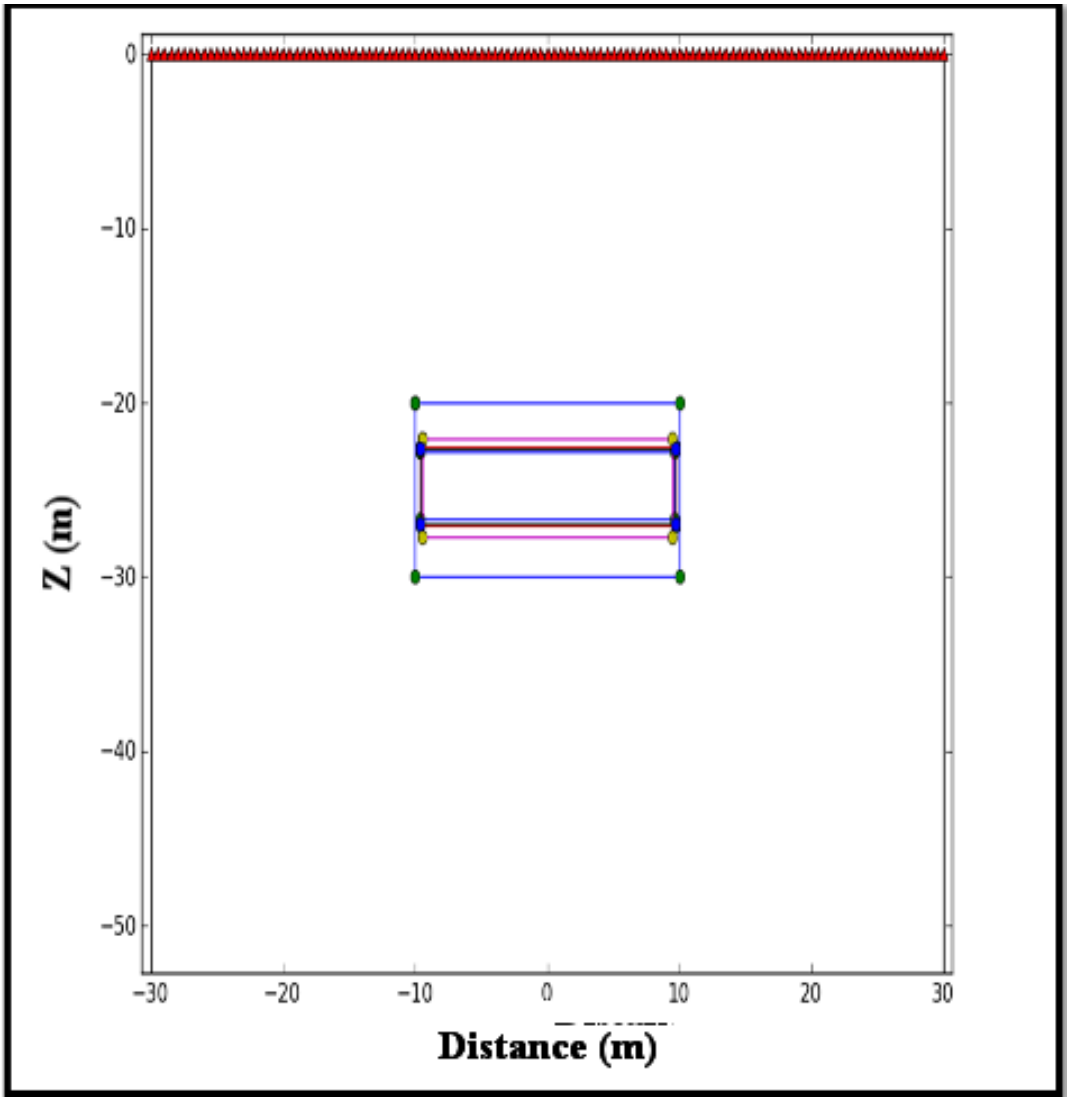

Figure 4.14: Diagram of target model (green corners and blue lines), with the eight models with lowest misfit in this suite.

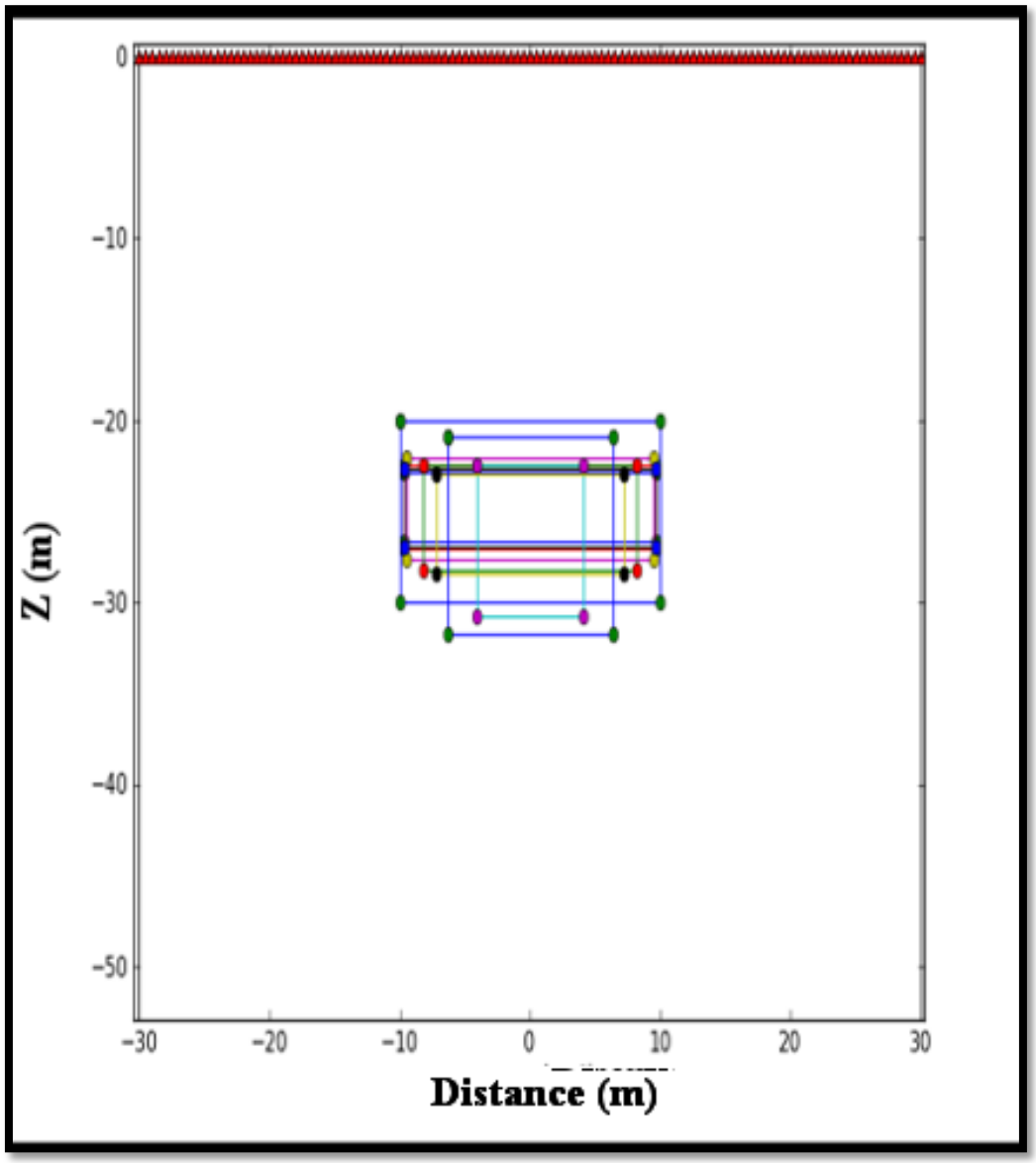

Figure 4.15: Diagram of target model (green corners and blue lines), with the eight models with lowest misfit in this suite.

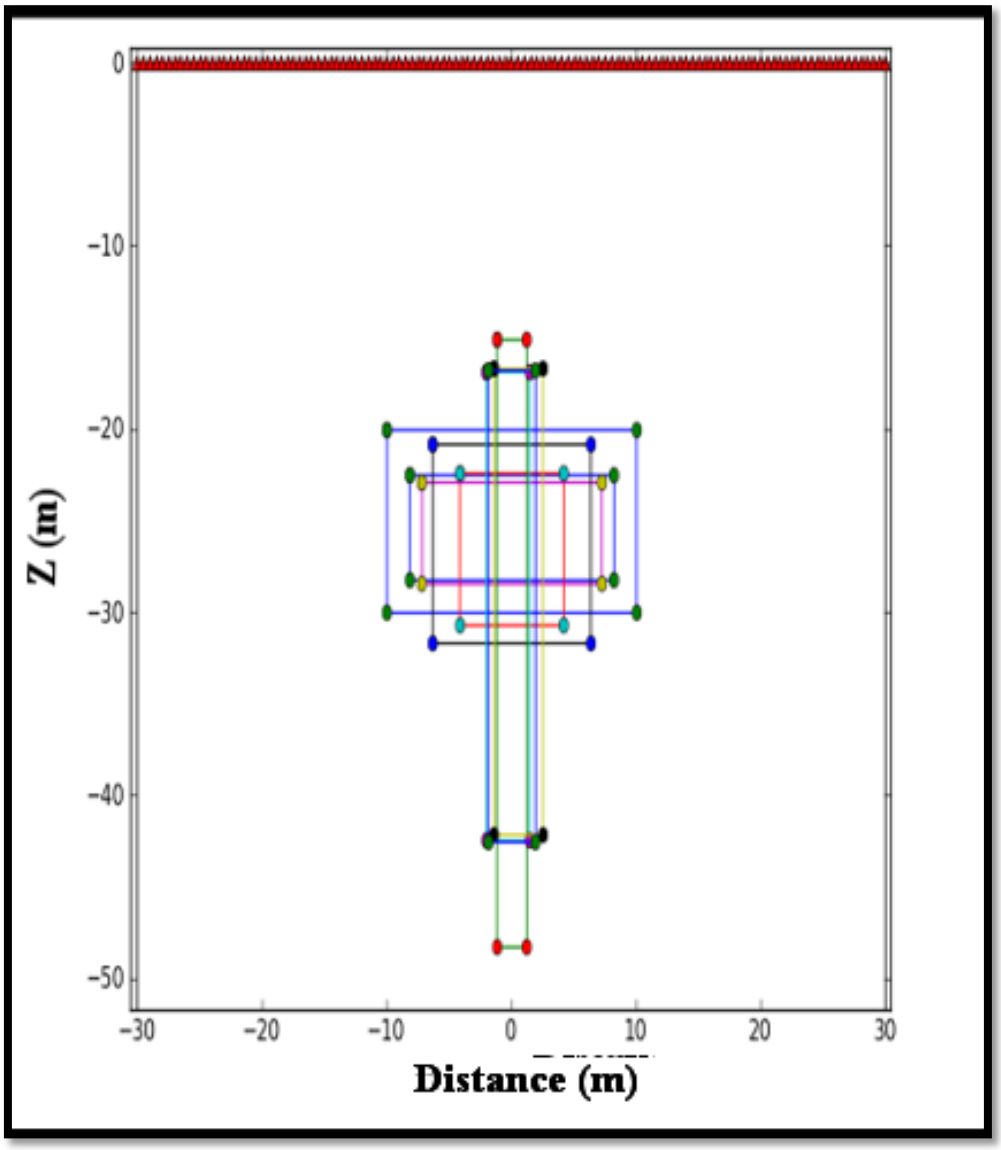

Figure 4.16: Diagram of target model (green corners and blue lines), with four highest misfit models and four middle misfit models.

For the reasons stated above, these results are quite promising. However they needed to be viewed with mixed success. This is apparent in Table 4.1, where in nine out of the twenty inversion results the values of x and z are rearranged, to have the first corner, X_1, Z_1, located in the bottom right instead of the top left. This is caused by the fact that there are no physical constraints during the inversion, besides the boundaries of the inversion domain, which allows the positions of the corners to move around freely. For the four-sided model this isn't a problem because the way in which the corners move still creates a box shape and the corners are still numbered in a clockwise fashion which is critical for computing the predicted data by the method described in Section 4.1. However this is most likely a consequence of the way in which the models are parameterized, with the sharing of coordinates. It seems that this created an unintended constraint which prevented the four-sided models from turning themselves inside out. This condition is solely applicable to the four-sided model as will be seen subsequently in results for the five-sided inversions.

4.3.2 Unconstrained five-sided inversions

The results for the five-sided model with no constraints had an average misfit value of 60.7 over the twenty inversions. Of the twenty, none achieved a misfit low enough to trigger the stop parameter. The lowest misfit achieved was 5.5 and the highest misfit was 189.0. When compared to the results of the four sided object these values are fairly similar. The majority of the constructed models have misfits that fall below the average line; however there are less models with misfits near the average and there are more high values in this set than the four-sided runs, seen in Figure 4.17.

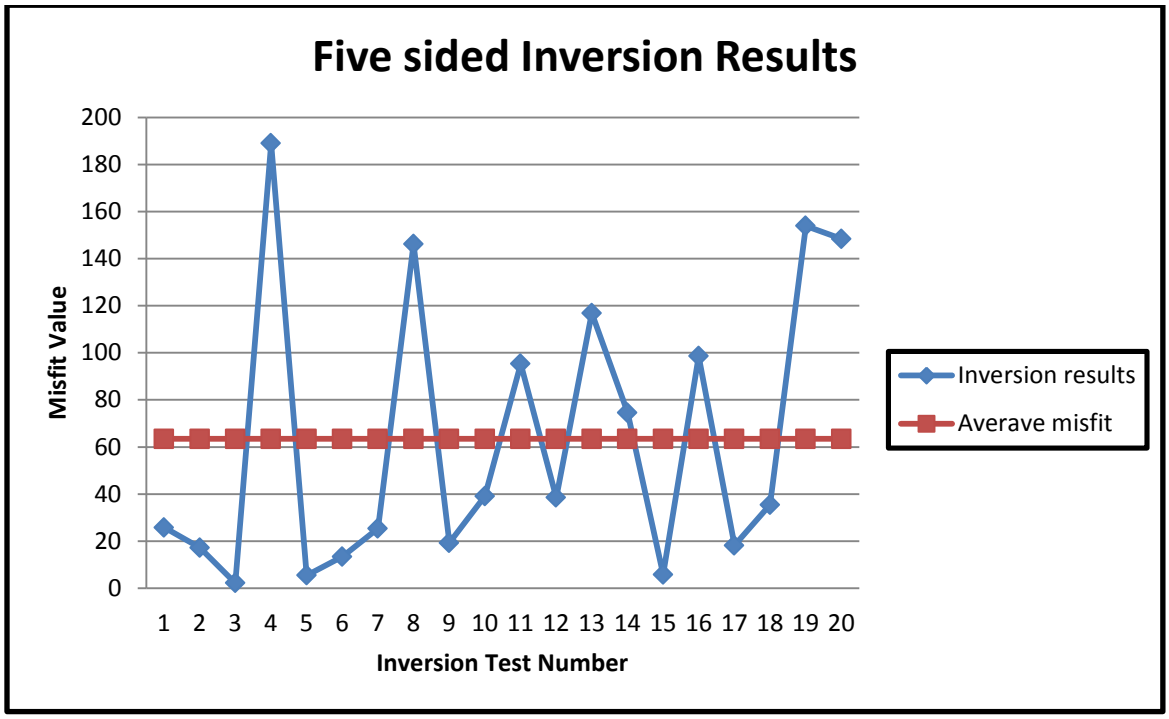

Figure 4.17: Line plot of inversion results returned from tests on five-sided model without constraints.

Figure 4.18 shows the original five-sided model with green corners and the constructed inversion model with blue corners. The synthetic observed data is the blue curve, and the predicted data are represented by the red +. The blue cornered shape is the inversion result with the highest misfit returned from the 20 runs, with a misfit of 189. Similar to the four-sided inversion results, the predicted data for this model are very similar to the synthetic data; there is little difference in shape, placement and amplitudes. However the model itself is turned inside out. Table 4.2 shows the values from which the corners of this model are derived. It shows that a similar rearrangement of the order of the corners has happened for this model, however since there are now five sides the object

doesn’t retain its intended shape. Even though this would be considered a successful inversion in terms of misfit, it cannot be overlooked that the model is turned inside out. Therefore this would be considered a failure. Of the twenty inversions done, only three retained acceptable shapes. These were test numbers 6,12 and 20. The average misfit for these inversions was 50.625, with the highest misfit 148, and the lowest 13.4. These three results would all be considered successful inversions. Figures 4.17,18 and 19 show the results of inversions 6,12, and 20 respectively. Examining these figures and Table 4.2, it is clear that they managed to retain an agreeable shape, even with the corners turned around, by placing the point which would be the “peak” of the house either inside or at the bottom of the object instead of at the top. This allowed the points to be in a proper order when evaluated in a clockwise fashion. Overall the five-sided inversions were successful in finding models that predicted the data well in every case, but in the majority of these cases the models were unacceptable due to their unacceptable shape. Therefore it is clear that, without constraint-handling methods, further inversions using these parameters or increasing the number of sides would inevitably end in failure.

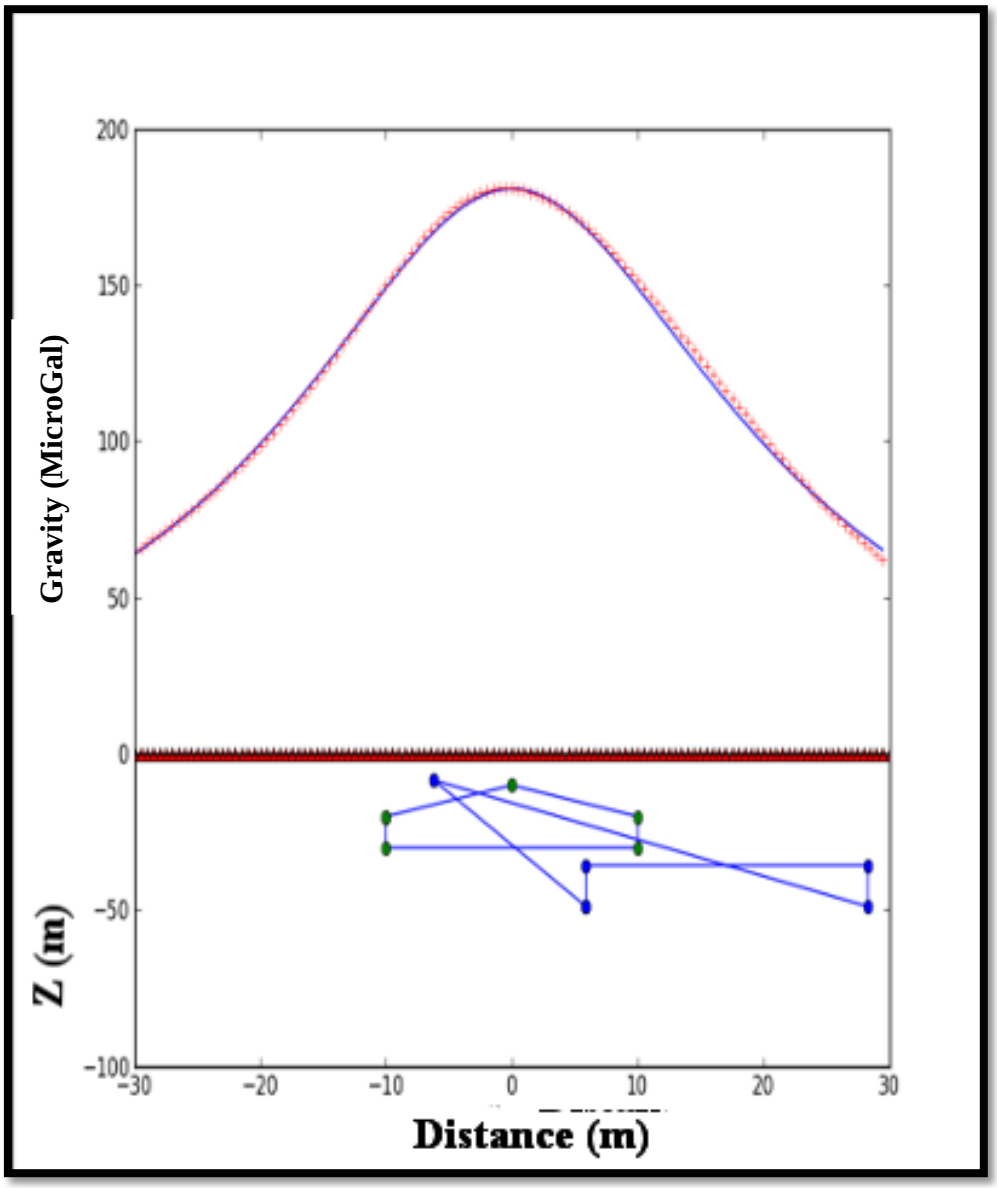

Figure 4.18: Diagram of the predicted model (misshaped object with blue corners) and data (red + curve), for the highest inversion result with misfit 189, shown with target model (house shape with green corners) and synthetic data (blue curve).

Table 4.2: Inversion results of tests performed on five-sided models without constraints.

Test #	Misfit	Time Taken(min)	X1	X2	X3	Z1	Z2	Z3	Rho
1	25.79	20.68	-15.7	1.6	8	45.8	9.3	34.4	1871.8
2	17.22	21.3	-13.5	-0.1	14.7	30.9	12.7	27.32	1930.2
3	2.2	21.1	7.95	0.54	-8.77	11.35	38.34	15.24	1714.5
4	189	19.2	5.9	-6.2	28.3	48.8	8.4	35.6	1856.2
5	5.5	20.1	4.9	7.7	-4	35.4	27.8	14.68	1989.1
6	13.4	17.1	5.63	-1.69	-5.99	36.14	21.77	15.06	1864.31
7	25.36	19.9	8.8	-2.6	-5.6	10.3	45.5	13.7	1506.2
8	146.2	19.4	-18	0.6	10.9	73.8	7.24	45.9	1688.7
9	19.3	19.5	-11.9	0.2	12.2	65.5	9.1	42.1	2324.7
10	39.14	19.5	-17.2	-17.1	2.88	8.5	47.6	35.4	2069.3
11	38.5	19.3	-23.2	2.05	12.5	49.9	8	36	1187.78

12	38.5	19.7	6.16	2.5	-6.8	18.45	37.7	15.4	1846.2
13	116.9	19.2	8.18	4.98	-13.8	13.1	34.5	15.7	1664.6
14	74.6	19.3	-22.2	-19.9	2.8	8.3	59.2	38.8	2172.23
15	5.8	21.2	6.24	-16.2	-5.6	31.8	34.1	13.5	1329.9
16	98.6	21.4	4.46	-7.8	-3.6	30.8	43.4	12.5	1588.6
17	18.2	21.5	7.4	-28.9	-7.3	34.7	33.7	12.5	956.9
18	35.45	21	-4.5	8	5.13	13.2	14.4	33.5	1609.8
19	153.9	20.6	3.2	-13.9	-3.7	42.87	30.2	13	1941.2
20	148.4	20.6	4.2	0.6	-3.8	18.4	53	12.7	1747.5
average	60.708	20.134							1722.746

Table 4.2 continued

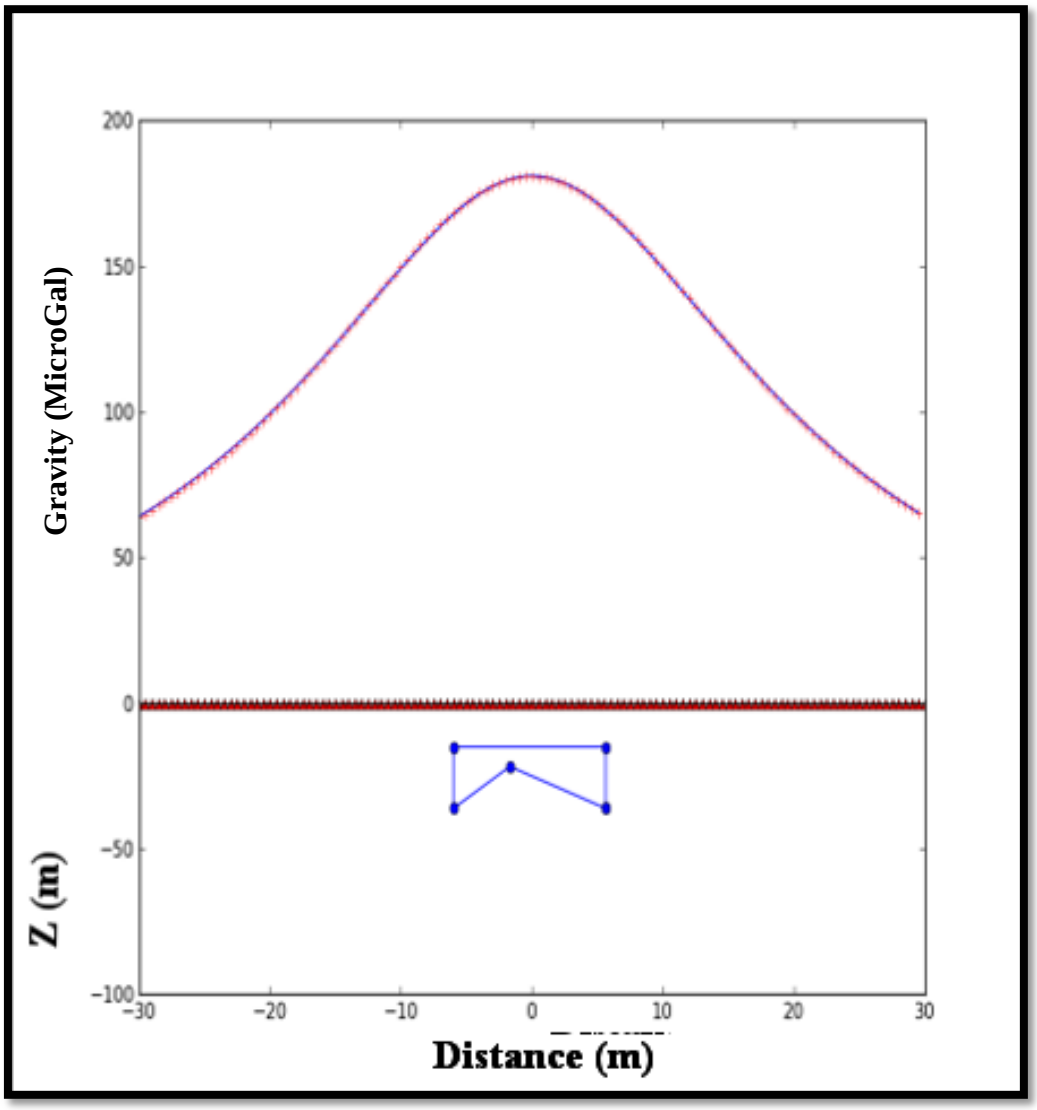

Figure 4.19: Model and predicted data result for test number 6 (red +), with synthetic data (blue curve).

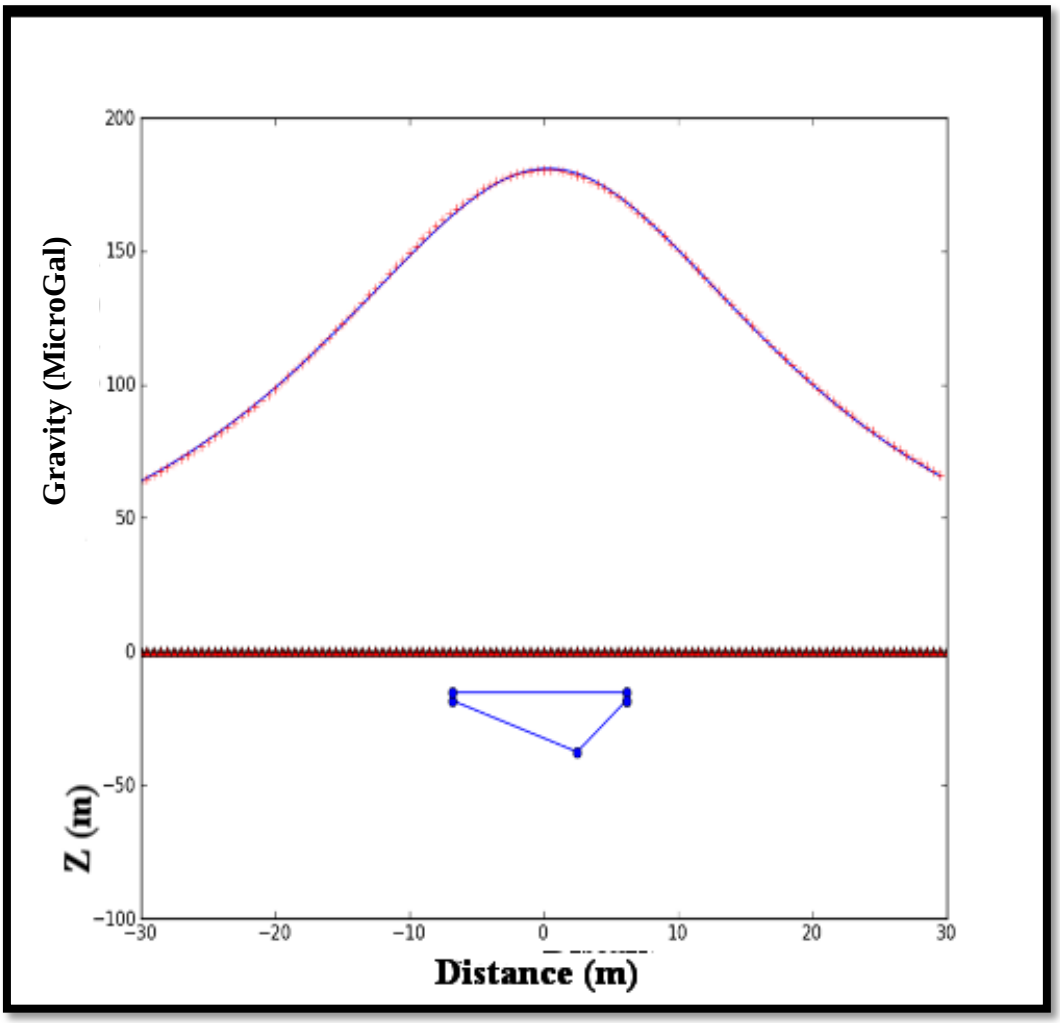

Figure 4.20: Model and predicted data result for test number 12 (red +), with synthetic data.

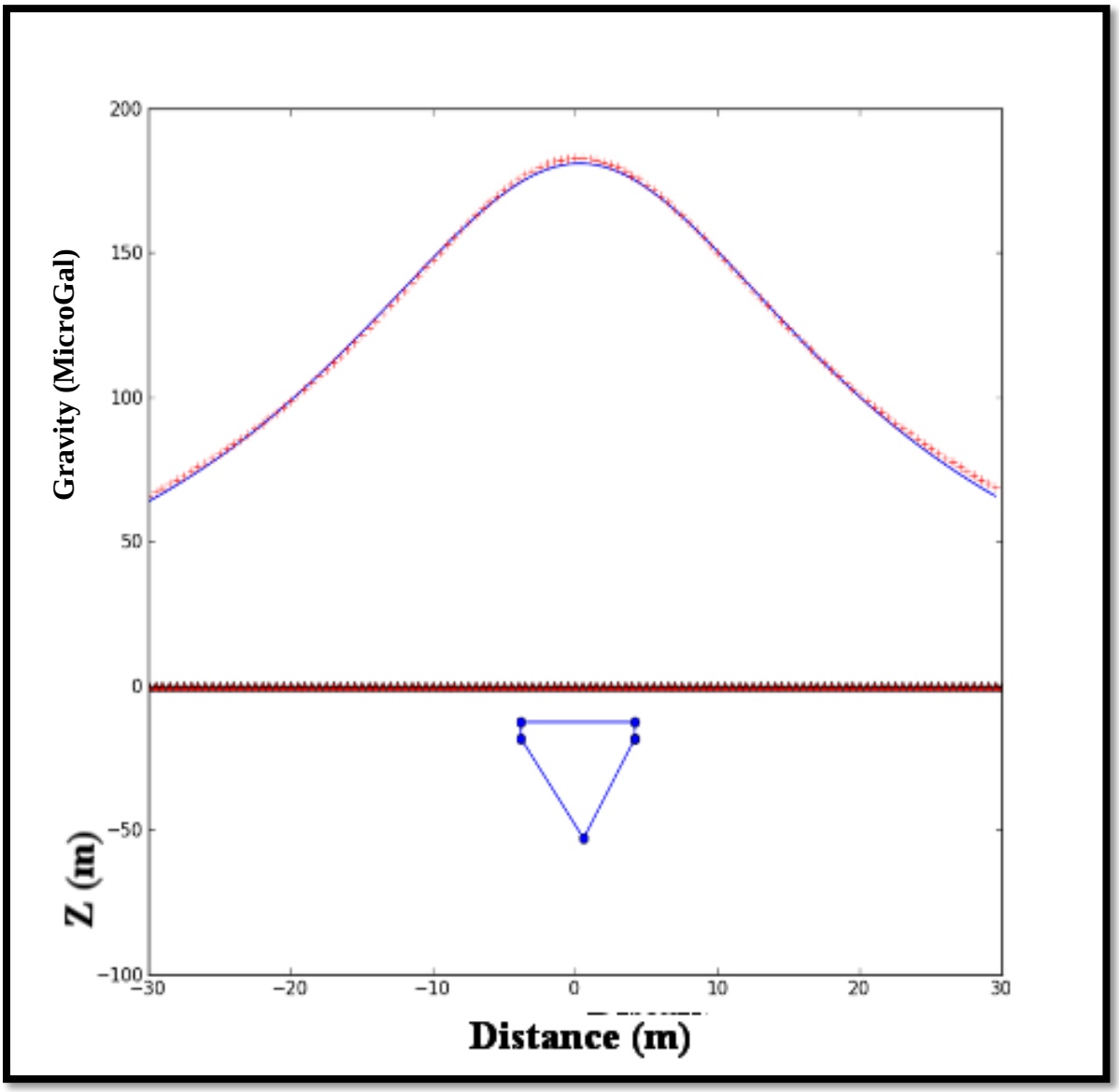

Figure 4.21: Model and predicted data result for test number 20 (red +), with synthetic data.

4.3.3 Constrained four-sided inversions

The inability of the models to remain in a proper shape prompted an investigation of constraint-handling methods, which led to the creation of the "Corner Switching" method described in Section 4.2.4. The corner switching method operated as its name implies. When unacceptable model orientations are detected the corners are swapped, restoring an acceptable shape. With this method applied to the inversion algorithm the resulting models have a large improvement in overall misfit value. The highest misfit recorded during this set of twenty runs was 86.0 and the lowest was 0.1, leading to an average misfit of 22.4. The average time to completion for these inversions was 54.3 minutes, which is slightly longer than the previous average of 50.3 minutes. This increase in time however is more likely due to the decrease in the number of stop functions achieved, only 2 in this set, rather than the constraints slowing down the algorithm.

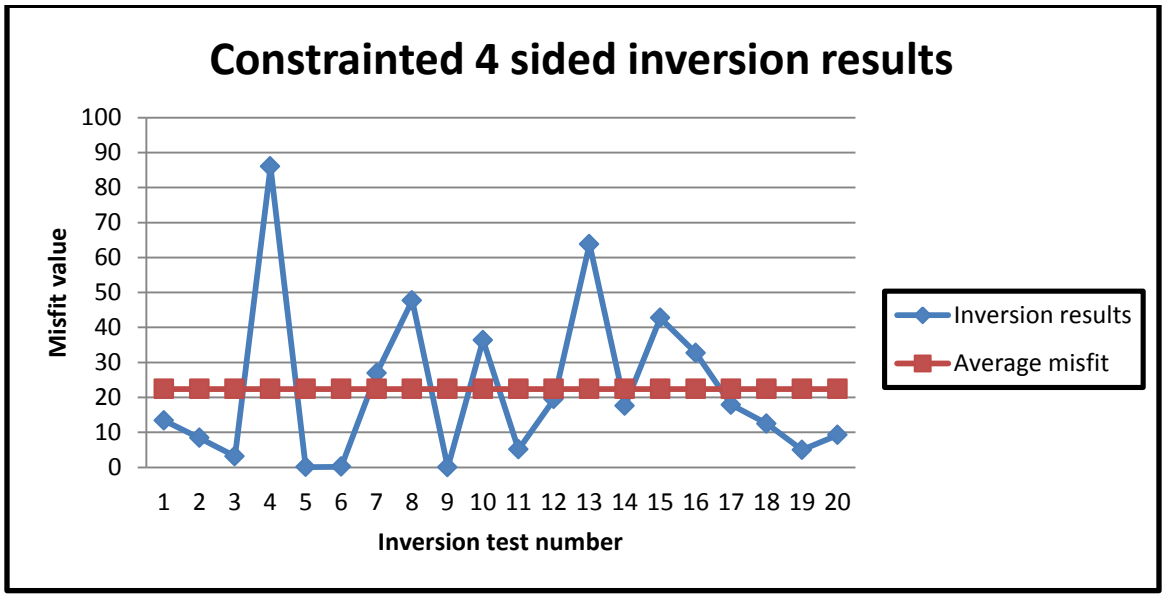

Figure 4.22: Line plot of inversion results returned from tests on four-sided model with corner switching constraints.

Figure 4.22, a plot of the inversion misfit results and the average misfit, shows that the majority of the values lie below or around the average line, and that there are three higher outliers, which are very similar results to the unconstrained four-sided models (Figure 4.9).

Figure 4.23, a plot of the predicted data and model of the inversion result with misfit 86.0, demonstrates that the model with the highest misfit for this series of tests is indeed a successful inversion.

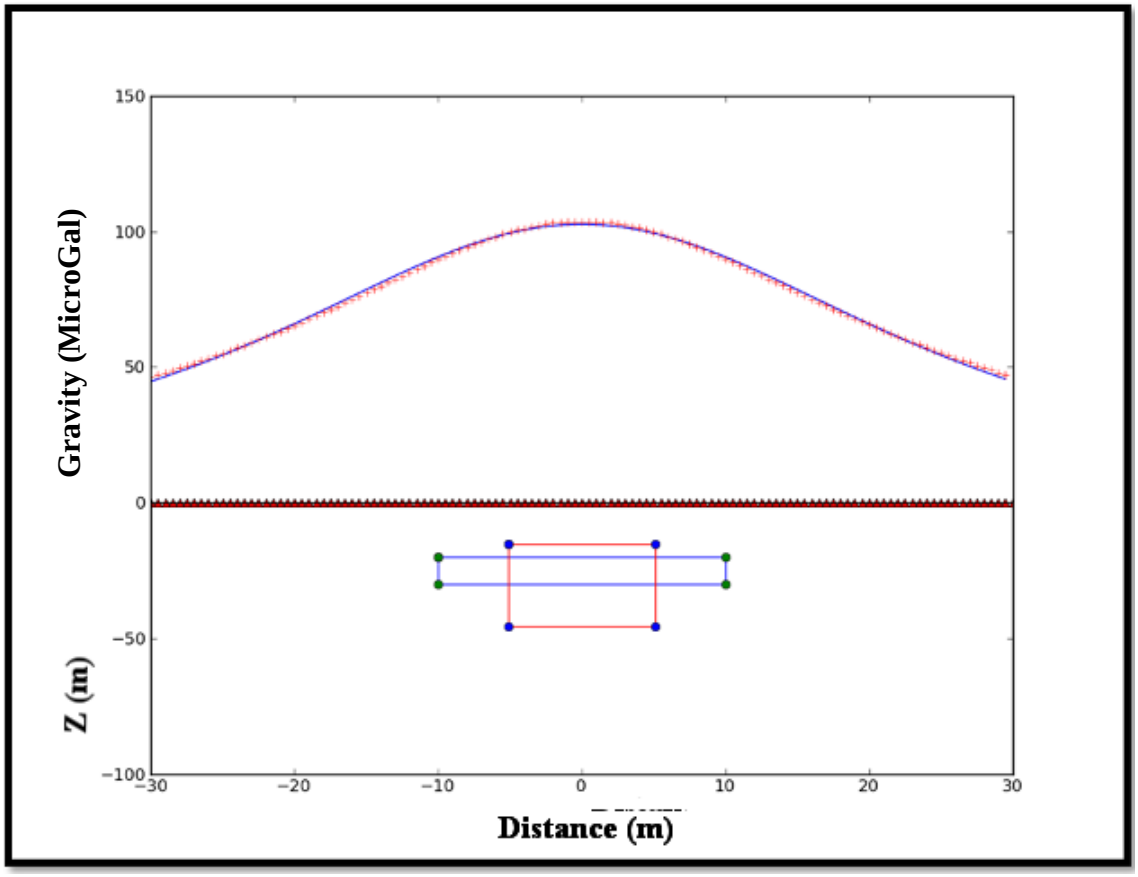

Figure 4.23: Model and predicted data result for test with misfit value 86.0 (red +), with synthetic data (blue).

Further evidence of this success is shown in Figure 4.24 which shows the difference between the observed and predicted data for this model. It can be seen that the difference between the observed and predicted data fluctuates along the observation locations, fitting better in some areas and worse in others. However it must also be noted that the maximum amplitude of these fluctuations never exceeds 2.0 microGal and is generally confined to a difference of less than 1 microGal.

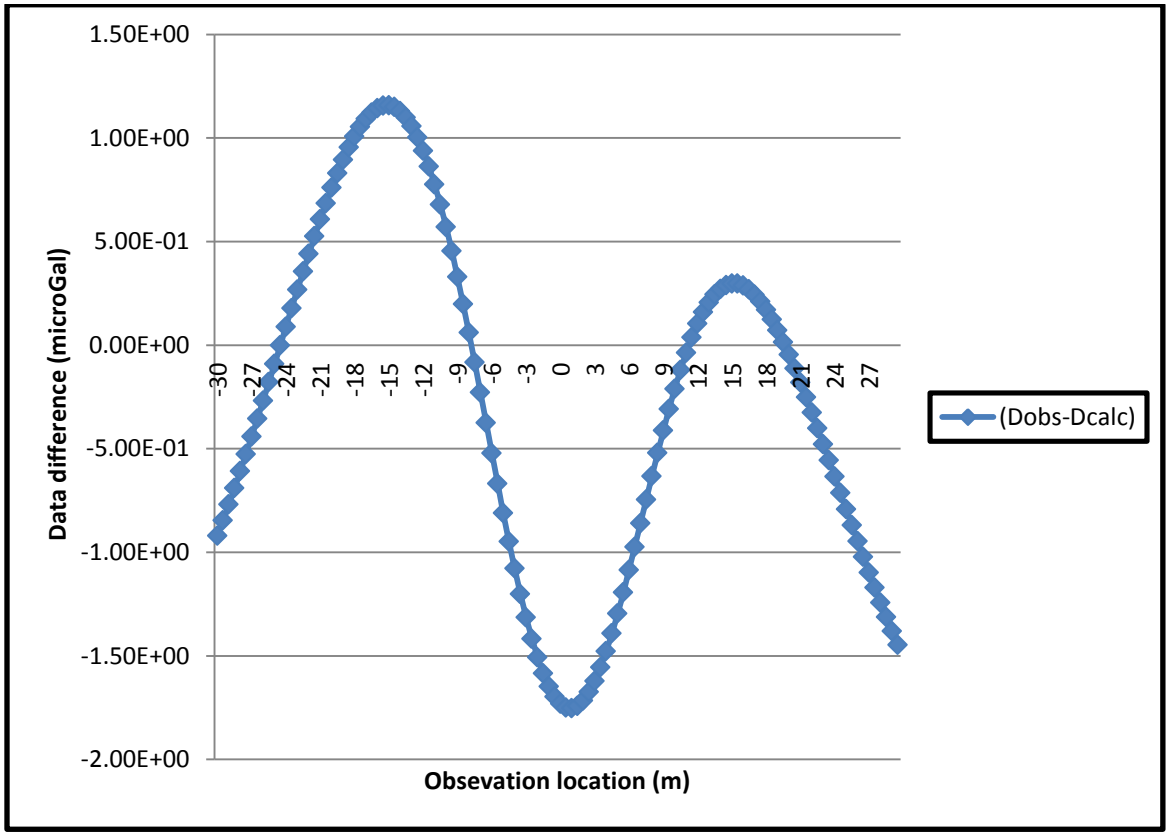

Figure 4.24: Data difference plot for model 4, with misfit 86.0.

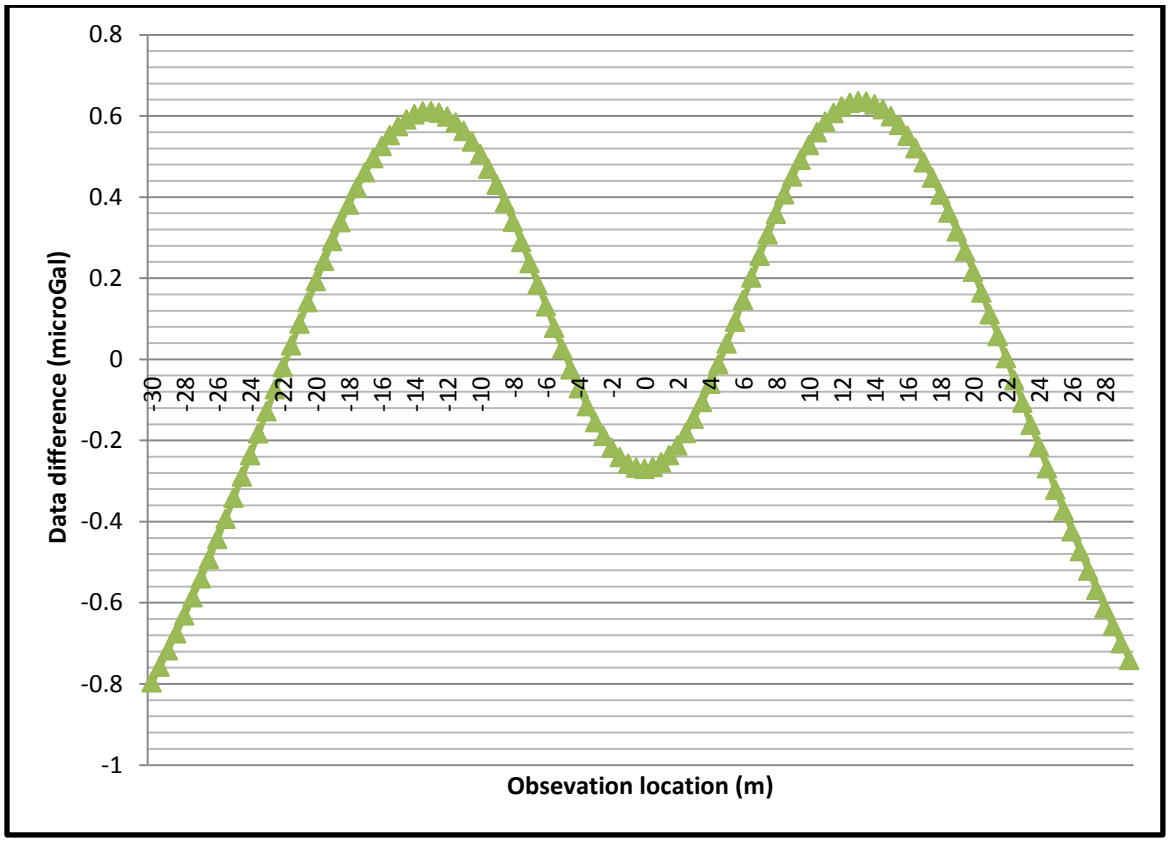

Figure 4.25: Data difference plot for model 12, with misfit 19.5.

Figure 4.25, a data difference plot for the model with a misfit closest to the average value of 22 with 19.5, shows very similar characteristics to the previous example. This plot is more symmetric, indicating that the positions of the observed and predicted data are more in sync, and the model is more likely to be symmetric about the origin. Another detail of this plot to note is the amplitudes of the difference. Here they are always less than 1.0 microGal, and the majority of the values fall between 0.5 and -0.5 microGal.

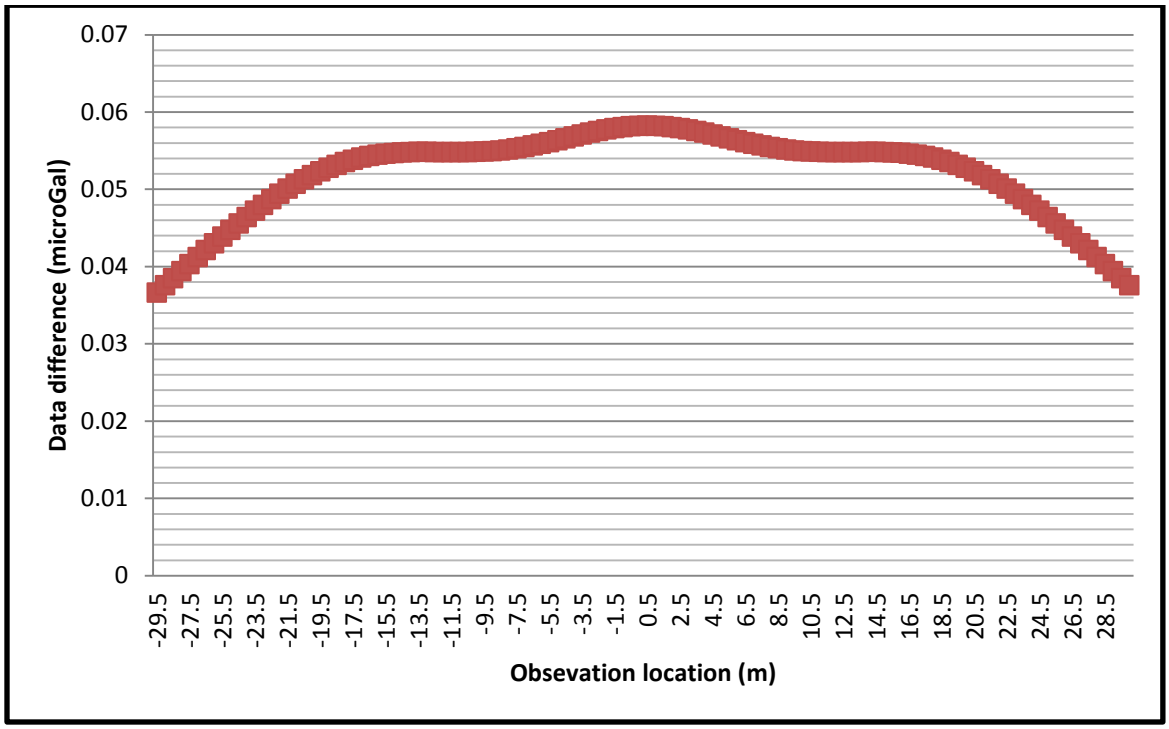

Figure 4.26: Data difference plot for lowest misfit model returned, with misfit 0.1.

Figure 4.26 shows the difference between the observed data and the predicted data created from one of the models which achieved a misfit value of 0.1 microGal. The lack of fluctuation in this plot compared to the previous examples indicates that the shape of the predicted data curve is very close to the observed data: also the ordinate values show that the amplitudes of both data sets are extremely close, the highest difference reaching only 0.06 microGal. This above all shows the success of this inversion set. However one thing remains to be verified, namely the success of the constraint handling methods applied.

Table 4.3 shows the information for all twenty inversions with the corner switching constraint for the four-sided object. It can be seen that the corners are no longer rotating and all the models have the first position in the top left corner of the model, as was intended. The corner switching method has fixed the previous problem, and the outcome is an algorithm that performs extremely well on a four-sided object.

Table 4.3: Inversion results of tests performed on four-sided models with corner switching constraints.

Test #	Misfit	Time Taken(min)	X1	X2	Z1	Z2	Rho	convergence iteration
1	13.4	57.5	-12.36	12.53	10.84	47.62	232.44	
2	8.4	58.53	-5.83	5.82	21.7	30.94	1897.72	
3	3.17	63.47	-7.08	7.09	22.91	28.58	2519.3	
4	86	63.7	-5.08	5.13	15.07	45.42	700.29	
5	0.1	14.4	-9.58	9.59	22.95	26.55	2885.2	1351
6	0.17	51.24	-10.04	10.06	19.16	31.24	826.23	
7	26.95	53.29	-4.66	4.65	18.89	36.32	1284.12	
8	47.7	57.9	-2.13	2.13	18.2	38.7	2418.7	
9	0.1	5.3	-10.07	10.07	19.74	30.29	940.79	465
10	36.36	57.5	-2.77	2.84	19.01	36.99	2079.1	
11	5.1	57.2	-7.56	7.59	20.28	31.84	1161.5	

12	19.5	58	-3.48	3.47	21.09	33.17	2464.5	
13	63.8	58.6	-5.99	6.23	15.53	43.1	634.85	
14	17.6	56.2	-4	4	20.8	33.1	2092.77	
15	42.7	70.7	-2	1.9	18.8	37.1	2882.65	
16	32.7	60.9	-2.7	2.8	19.48	35.79	2295.4	
17	17.9	62.7	-3.16	3.19	21.36	32.63	2886.76	
18	12.5	56.8	-5.15	5.18	21.13	32.24	1791.26	
19	5	67.1	-6.95	6.94	21.49	30.5	1623.98	
20	9.22	54.5	-6.25	6.26	20.68	32.17	1423.56	
average	22.41204	54.2765	-5.842	5.8735	19.4555	34.7145	1752.056	

Table 4.3 continued

4.3.4 Constrained five-sided inversions

Applying the corner switching method to inversions of five-sided models wasn't a simple matter of just adding the criteria developed for four sides to the five side code. In the four-sided algorithm, the corner switching method consisted of keeping track of only two possible infractions; if the left and right sides are switched and if the top and bottom are switched. This however is far too simplistic to tackle the complexities arising from the addition of just one extra point. Adding another point means that for every time one infraction is detected two other points need to be checked, for every point in the model, leading to a far more complex process. After several failed attempts at implementing the required changes, a working corner switching method was finally applied to the five-sided model algorithm. The changes involved creating an elaborate and complex set of criteria for checking and rearranging the corners when violations occured. The addition of the corner switching method rectifies the problems experienced in the previous examples as can be seen in Table 4.4 and Figure 4.27. Examining the table and figure shows that each model is an acceptable shape and is oriented correctly. Unlike previous results (shown in Figure 4.13) there is a large contrast in the majority of the models: some are similar in shape but there is no grouping or trend present.

Table 4.4: Inversion results of tests performed on five-sided models with Corner switching constraints.

Test #	Misfit	Time Taken(min)	X1	X2	X3	Z1	Z2	Z3	Rho
1	321.23	23.23	-10.83	-1.25	13.7	32.69	6.47	32.85	954.77
2	90.33	23.51	-8.43	5.61	7.74	17.47	15.27	26.22	1846.41
3	138.3	23.55	-9.99	-0.69	10.83	26.33	7.2	32.53	940.94
4	202.34	23.62	-11.32	-7.54	11.33	10.43	10.27	33.54	566.02
5	3.08	17.15	-6.34	1.92	6.2	15	13.23	31.81	1365.24
6	143.798	17.6	-11.49	-3.099	13.55	21.47	9.64	28.08	942.99
7	1.6	20	-10.98	-0.064	11.02	22.08	9.49	29.47	996.43
8	36.58	20.03	-9.56	-1.27	9.58	8.77	8.61	41.92	494.11
9	48.95	22.27	-11.18	2.95	10.55	14.37	8.85	33.45	633.52
10	432.31	22.19	-22.1	0.49	21.91	19.7	12.08	20.19	1460.99
11	1.12	22.33	-6.83	-0.177	6.867	16.88	12.088	31.308	1313.496

12	275.967	22.82	-4.415	1.533	3.455	22.537	7.545	36.554	1872.11
13	492.87	23.92	-7.19	4.49	6.89	9.66	7.07	49.47	575.397
14	3.66	24.166	-3.207	-0.677	3.18	15.78	15.469	30.65	3169.96
15	4.75	20.31	-9.11	0.37	9.07	12.86	9.77	35.69	689.42
16	607.55	20.25	-3.59	-3.177	5.21	20.06	6.37	40.55	1340.57
17	85.15	23.38	-9.76	-0.54	10.81	26.54	8.802	30.39	1164.31
18	47.73	23.42	-2.6	1.67	2.33	18.85	13.02	30.29	4267.17
19	55.17	24.16	-12.23	0.22	11.99	23.77	14.72	24.2	2440.67
20	46.93	24.08	-13.02	2.07	10.13	26.68	12.11	26.93	1754.51
average	151.9708	22.0993	-9.2086	0.14195	9.3171	19.09635	10.4037	32.3046	1439.452

Table 4.4 continued

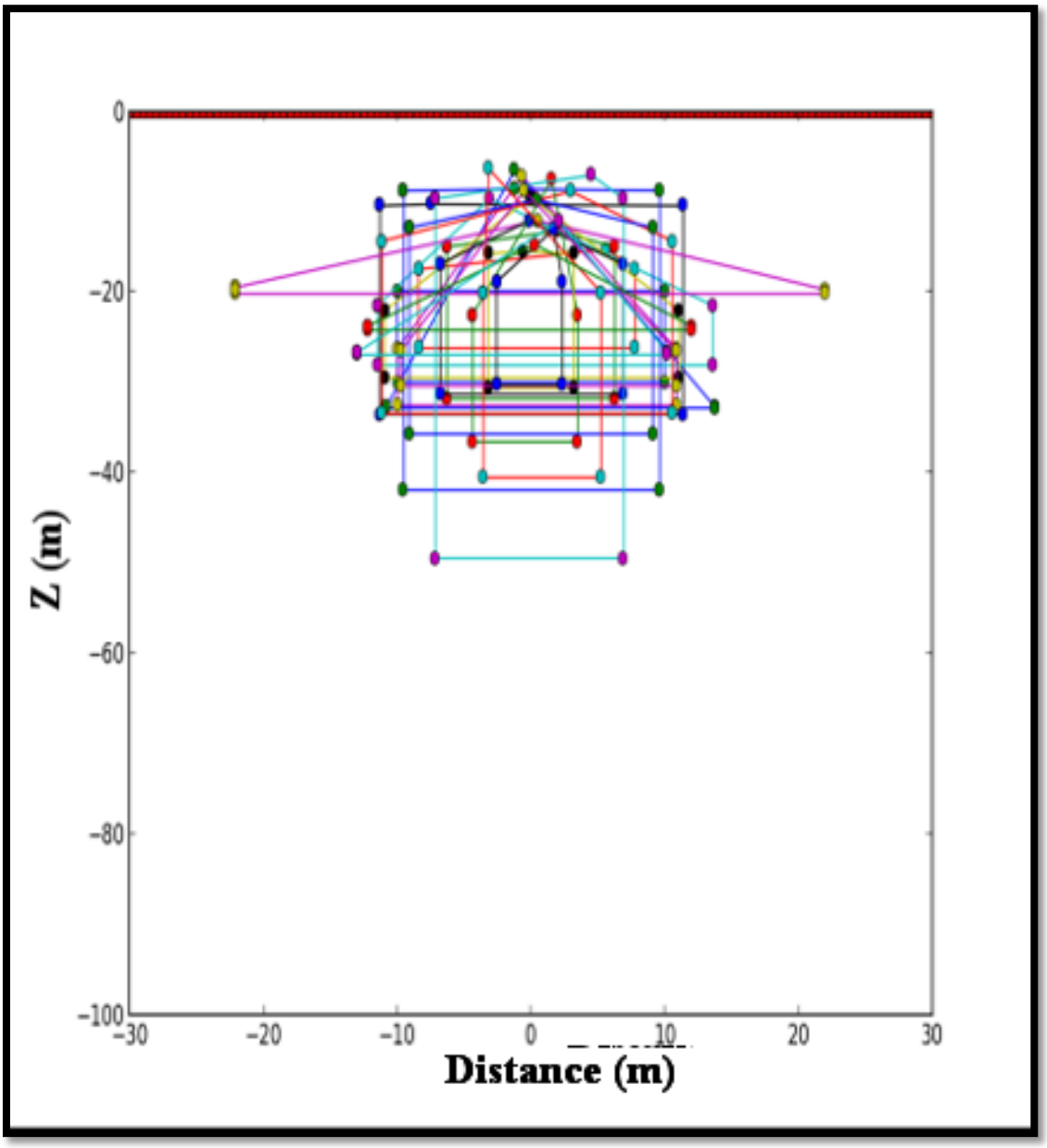

Figure 4.27: Plot of all model shapes returned from inversion of five-sided inversion with corner switching constraints.

The average misfit achieved for this set of inversions was 151.97, with a high misfit value of 607.55 and a low value of 1.12 (see Figure 4.24). Similar to the previous inversion sets the majority of the test runs have misfit values which lie below the average misfit, 11 out of 20. Four lie near the average line and the rest are high above the average.

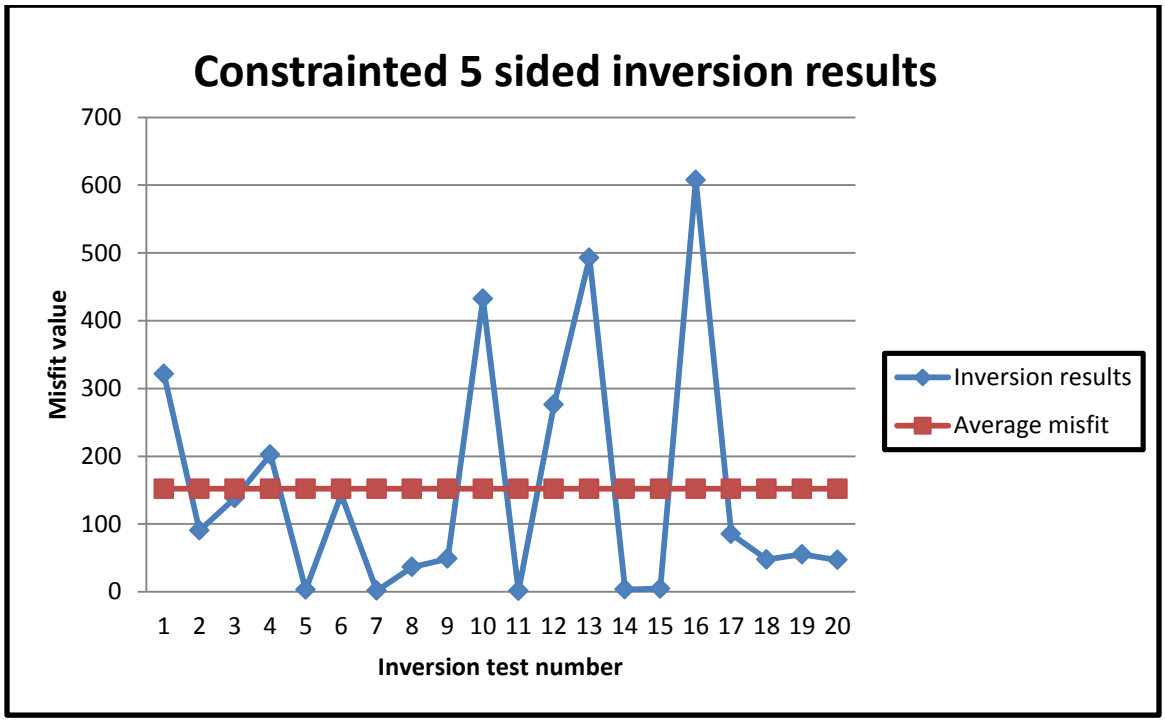

Figure 4.28: Line plot of inversion results returned from tests on five-sided model with corner switching constraints.

The average time to completion in Tables 4.2 and 4.4 are 20.13, and 22.10 minutes respectively. These times are clearly much lower than those of Tables 4.1 and 4.3, which had averages of 50.35 and 54.28 minutes. This is because a different type of

file, a Cython file which allows Python to run faster, was used for these two cases. The main thing to note about these times is that in the non-constrained five-sided inversions the average time to completion was 20.134 minutes, and the average time to completion for the constrained five-sided inversions was 22.09 minutes, a difference of just 2 minutes. The significance of this is that the corner switching method adds about two minutes to every inversion of this type, not a significant amount when weighed against the overall improvements it brings. The model with the highest misfit recorded during the constrained runs was 607.55. It has been demonstrated previously that models with a misfit as high as 226 are still considered successful inversions. Figure 4.29 shows the observed and predicted data for this model. It can be seen that the peak of the predicted data is not quite lined up with the observed data, and has a higher maximum amplitude. The shapes of the curves are very similar despite this shortcoming.

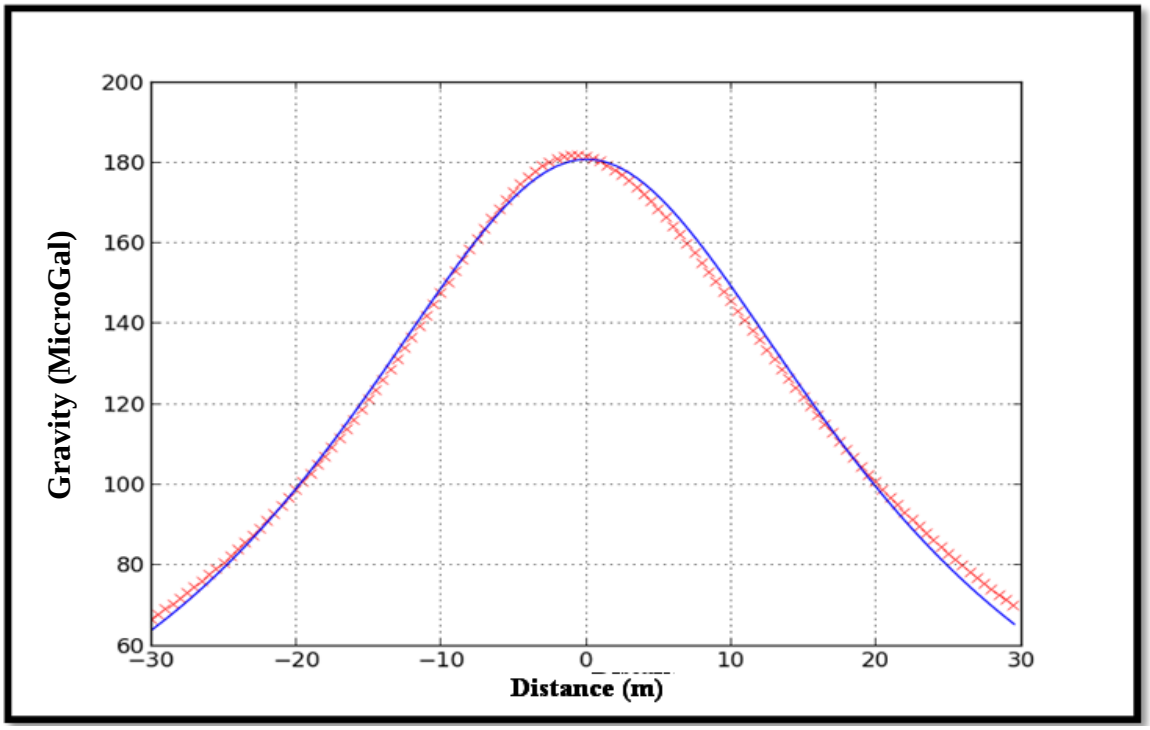

Figure 4.29: Observed (blue) and predicted (red+) data for model 16, with misfit 607.55.

Overall this would be considered a successful inversion; even though it has a large misfit value the similarities between the data curves and the fact that the model reproduces the original shape well deems it an acceptable result. The data difference plot for this model is shown in blue on Figure 4.30. The shape is similar to previous results with the fluctuation with observation location. The left peak is larger than the right due to the models middle point being located on the left (red in Figure 4.31). The maximum difference recorded is along the last observation locations of the right side, also caused by the models peak on the left side of center. Overall the majority of the difference is confined to between 2 and -2 microGal, an acceptable amount of difference when values are reaching upwards of 170 in the observed and predicted plots.

Also shown on Figure 4.30 are the difference plots for misfit 492 and 432 (red and green respectively). The difference plot of the model with misfit 492 is basically the same as model 607, except the middle point of the model is centered on the opposite side (green in Figure 4.31), causing the difference plot peaks to be opposite. The shape of the plot is generally the same for these two cases, but the placement and amplitudes are different since the models are oppositely oriented. The data difference for test model 10, which gave a misfit of 432, is seen in purple on Figure 4.30. Its shape is almost a mirror image of the previous models, which means the predicted data is lower than the observed data in all the places that the previous examples were higher. The main things to note however is it is confined to the same range of differences as the previous two examples.

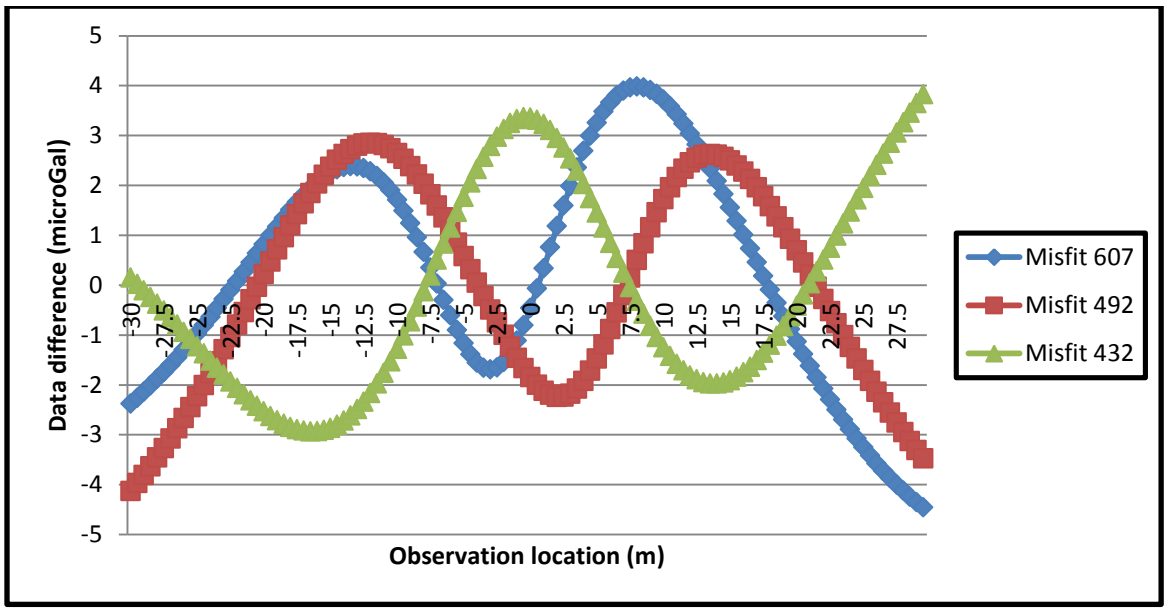

Figure 4.30: Data difference plot for models 16, 13, and 10 with misfit values 607.55, 492, and 432.31 respectively.

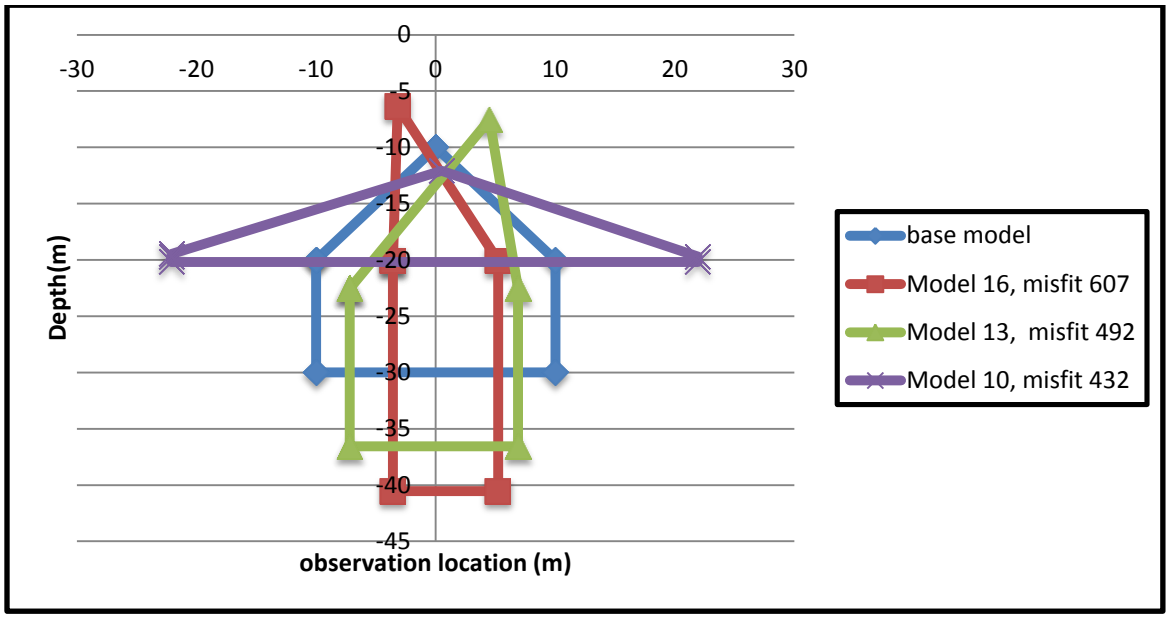

Figure 4.31: Model shapes for test numbers 16, 13 and 10, with target model.

Results from test number 6 were closest to the average value; Figure 4.32 shows the observed and predicted data for this inversion. The predicted data for this inversion are much closer to the observed data than seen in the previous examples. The model shown in Figure 33 has its peak off center to the left, which causes the slight misalignment. The highest amplitude in the data difference plot, Figure 4.34, is recorded near station locations -7.5 m and 5.0 m, which are associated with the misaligned peak. The amplitude in those areas reaches to almost +/- 2.0 microGal. The majority of the difference in the observed and predicted data is less than 1.0 microGal, which indicates a quality inversion result.

The three models with the best inversion results during this set were numbers 11, 7 and 5 followed closely by test numbers 14 and 15, with misfit values of 1.12, 1.6, 3.08, 3.66 and 4.75 respectively. Following the trend thus far, the best results come from models which are the most similar to the original, as seen in Figure 4.35. Models 11 and 7 have the center point located almost exactly at the origin, which allows the data to fit better since it isn't shifted to one side. Model 5s approach to this same problem was to suppress the shape down almost to a box shape with five points. Although this isn't as effective as aligning the peak with the origin, it is still much better than average and creates a very low misfit value. Figure 4.36, a data difference plot of these three models, shows just how close each set of predicted data lies to the observed. Model 5 has deviations between -0.25 and 0.4 microGal. Model 7 fluctuates between -0.3 and 0.15 microGal, an improvement over model 5, however not quite as impressive as model 11 which is confined between -0.2 and 0.15 microGal. These models are extremely close to the observed data, they are excellent inversion results.

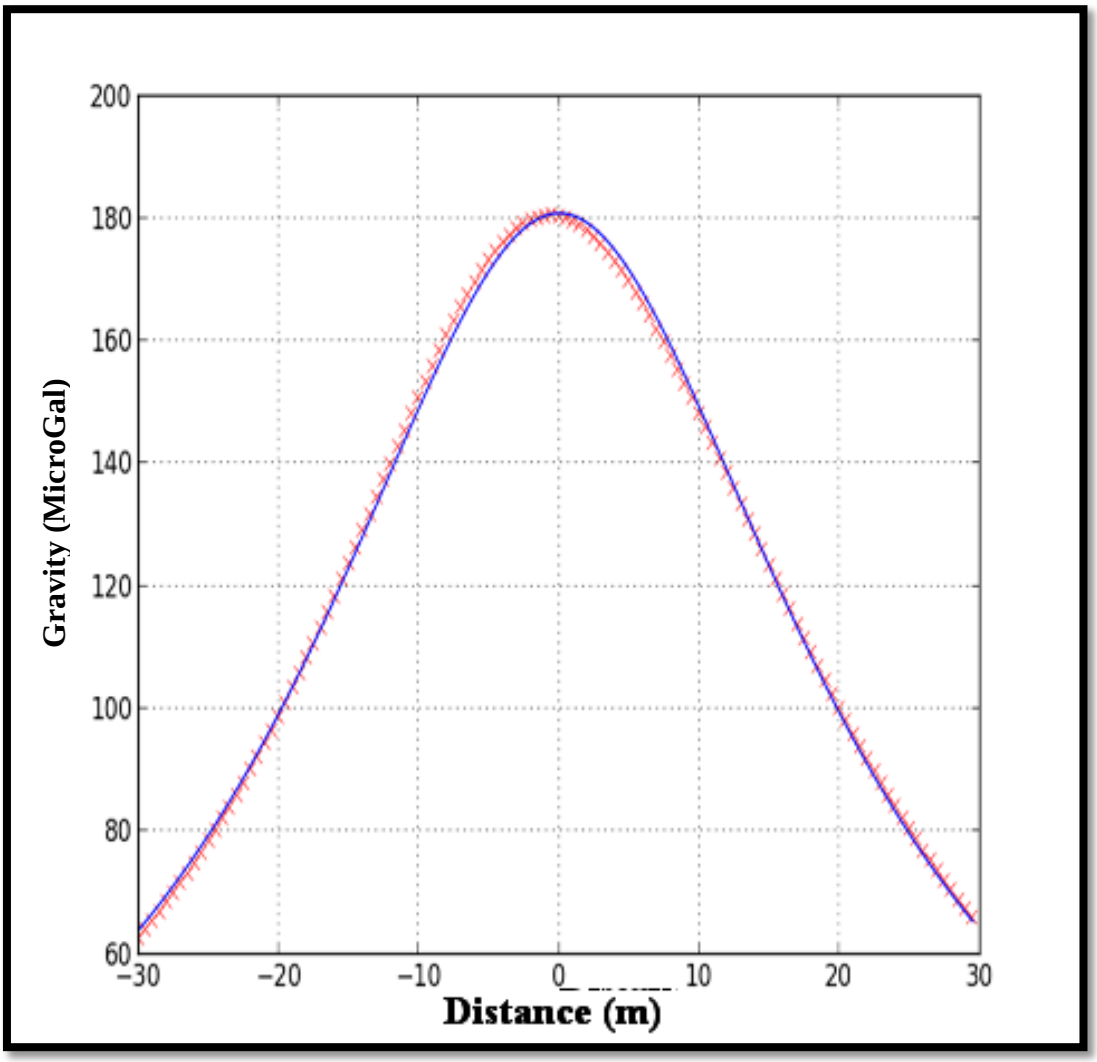

Figure 4.32: Observed (blue) and predicted (red+) data for model 6, with misfit 143.80.

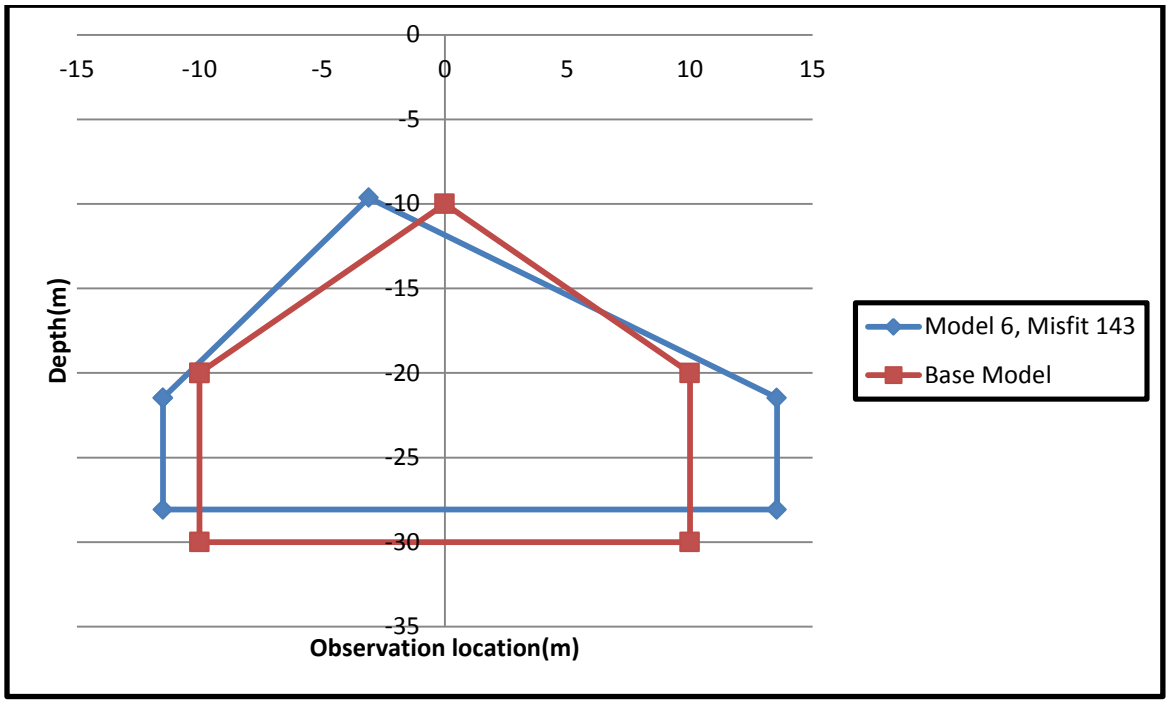

Figure 4.33: Model shape for test number 6, with target model.

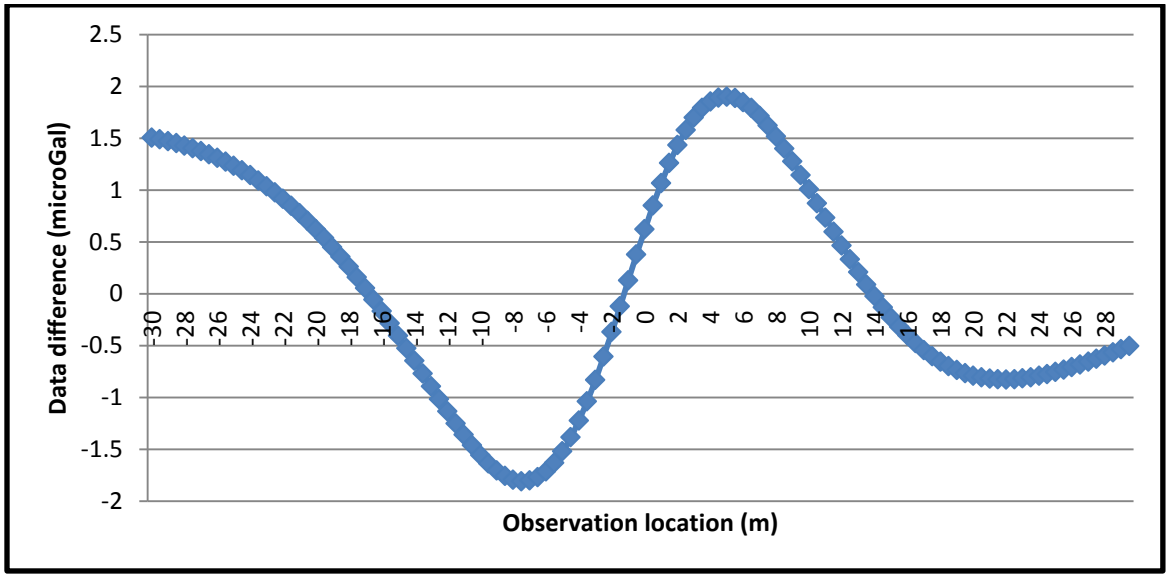

Figure 4.34: Data difference plot for model 6, misfit 143.

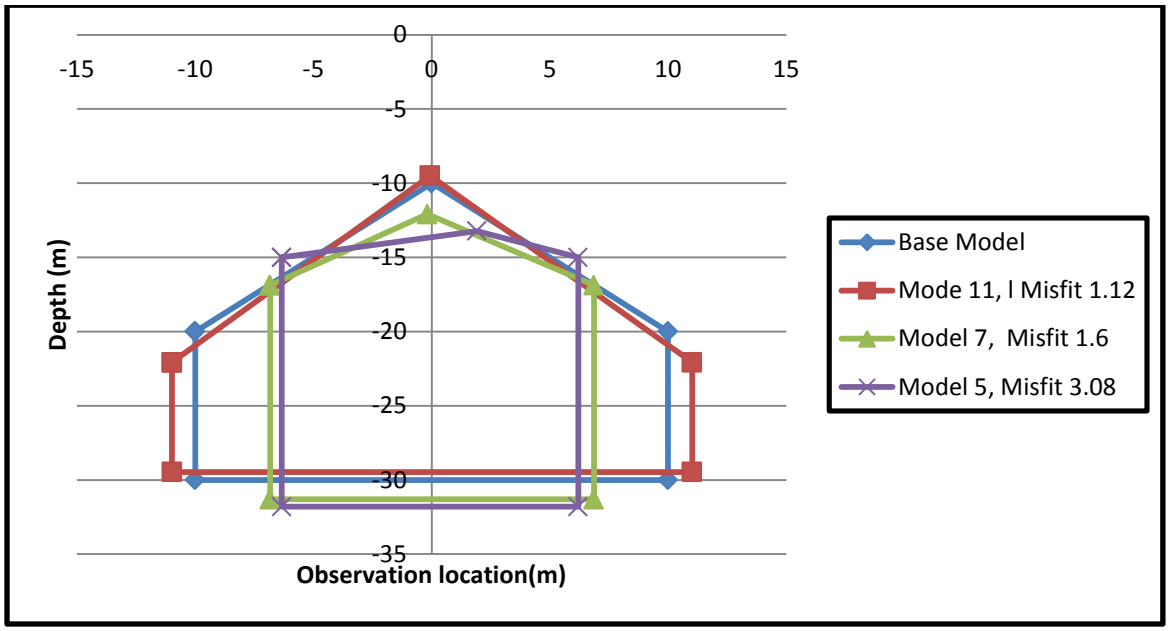

Figure 4.35: Model shapes for test numbers 11, 7 and 5, with target model.

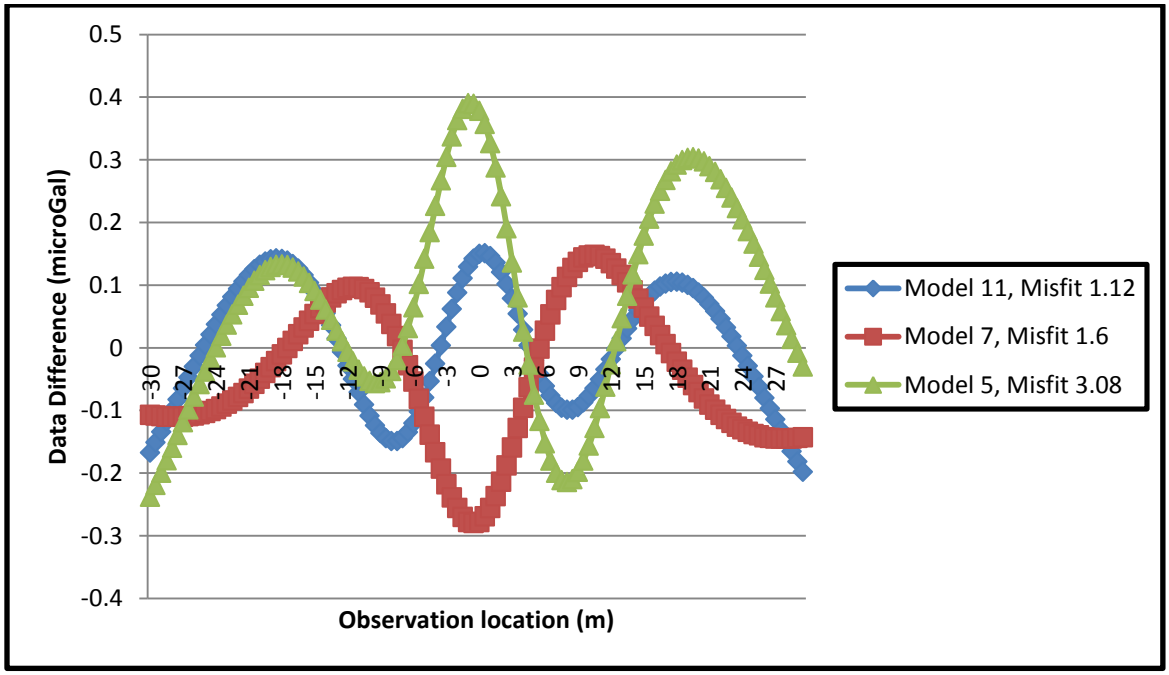

Figure 4.36: Data difference plot for models 11, 7, and 5 with misfit values 1.12, 1.6, and 3.08 respectively.

4.4 Conclusions

This chapter began by broaching the topic of forward modeling, and how it would be used as the evaluating agent of this inversion algorithm. The two inversion targets used to create the synthetic data for the test inversions where a 20m x 10m box shape and a house shape with the same dimensions as the box shape but which extends towards the earths surface in the middle. The chapter then touched on several topics required to convert the PSO software from minimizing functions to inversions of gravity data. The first requirement was to increase the number of parameters that could be used by the algorithm. Then using the “sides method” of parameterization, forward modeling can be used instead of function equations. These forward modeling calculations are then evaluated using a least squares misfit function.

Once these additions were applied to the algorithm, tests were performed on the four and five-sided objects with varied results. The results for unconstrained inversions of the four-sided model appeared to be quite successful, obtaining an average misfit of 41.4 with a range of misfits between 0.1 and 226. The only negative detail about these results was the tendency for the corners to rotate around, which isn’t an issue for his model but was quite problematic for unconstrained inversions of five-sided objects. Without constraints the inversions on the five-sided model failed to stay in an acceptable shape for the majority of the results. Only 3 out of 20 models were acceptable after the inversion was finished causing this to be a failed test set, and prompting the creation of the “corner switching” constraint method. The corner switching method consists of testing the shape of the model during an inversion by evaluating the position of the points that make up the

model. If for example, X1 is greater than X2 in an inversion of a four-sided model, this would mean that what should be the left side of the model is actually the right side. This would cause the corners to rotate, and thus the corner switching method would interchange the values of X1 and X2, preserving the proper shape of the model.

When the corner switching method is applied to inversions of the four-sided model, the results have a large improvement. The average misfit decreased by about half, from 41.4 to just 22.4 with values ranging from 0.1 up to 86.0. The average time to completion of this set increased by about 4 minutes, but this was considered more a factor of fewer stop functions encountered rather than the amount of time added by the corner switching method. The most notable improvement was that zero out of twenty models had corners that were switched around after the inversions were complete, signifying the success of the corner switching method.

The results of inversions of the five-sided models with the newly proven corner switching method were a great improvement over the previous set. Once again no models were left in unacceptable shapes after inversion, and over the twenty tests the average misfit achieved was 151.97, with the highest value being 607 and the lowest being 1.12. Then, even though the highest misfit was 607, it is still considered a successful inversion, as was demonstrated mainly by data difference plots which show that the fluctuations between the predicted and observed data for this test were less than 5 microGal for any given observation location.

Even though these were mainly successful inversions, improvements need to be made to the algorithm in order to move towards models with greater number of sides, and

to be more flexible overall. This was apparent when the transition from four-sided to five-sided models was attempted with the sides method. The downfall of the sides method is that it becomes problematic to parameterize objects with more than four sides; every shape would have to be symmetric, and have shared x and y coordinates, which is unacceptable when attempting to create complex objects. To counteract this deficiency a new method for model parameterization was created, henceforth known as the “points method”, to deal with the impending problems related to defining larger, more complex targets.

Chapter 5 : Constraint Handling and Points Methods

This chapter will further investigate the creation of an inversion algorithm by first highlighting the areas of the coding which remained unchanged. These areas include the forward modeling and misfit function. The chapter then details the differences between the sides method and points method of model parameterization. This leads into descriptions of the inversion target models used in this chapter. The new methods for constraint handling are then described, namely the penalty function and the rejection of infeasible solutions (RIS) methods. The results section discusses the outcome of the tests conducted with the penalty function and the RIS methods using the sides method. Later, once the tests have determined which method is better suited for the algorithm implemented here, the results of a six-sided inversion with the “points” method will be detailed. (The relevant source code can be found in Appendix B, along with all results not explicitly shown in this chapter)

5.1 Forward modeling methods and misfit function.

This chapter introduces the points method of model parameterization, along with several changes to the inversion algorithm. Some things however did not change, and operate nearly identically to the implementation used to generate results described in Chapter 4. The things which remained unchanged include the misfit function, the target models and the forward modeling technique. The misfit function remains as a least squares misfit, and performs exactly the same in every example except those which use a penalty function (Section 5.3). In these cases the penalty is added to the calculated value of misfit. The

target models also remain unchanged; this chapter will however introduce an additional six-sided model. The forward modeling method performs exactly the same for the sides method and the points method. The only difference is the manner in which the points are generated. The outcome is identical, as can be seen in Figure 5.1, which shows two sets of synthetic data, one calculated using the code for the sides method and the other using the new code for the points method. Chapter 4 mentions the use of Cython files to speed up the inversion process. This technique is also used in this chapter for testing of the penalty function and RIS constraint handling methods. With the addition of the points method, a new method for optimizing the time used during inversion, called Parallel Python, is implemented. Parallel Python is a python module which provides a mechanism for parallel execution of Python code on systems with multiple processors or cores, and on clusters. The benefits of this method will be described in the results section of this chapter.

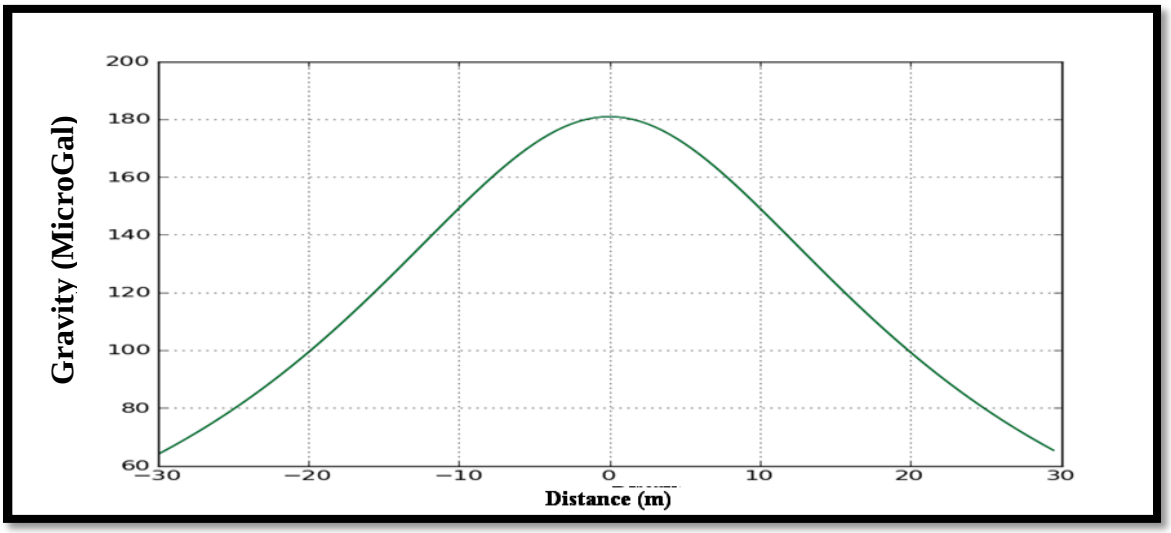

Figure 5.1: Plot of two identical sets of synthetic data calculated using code for the sides method and points method.

5.2 Model parameterization and inversion targets.

The points method was developed to overcome difficulties arising from creating models with more than four sides. Unlike the sides method, which used the intersection of X and Z values to determine the corners of each polygon, the points method uses a pair of X and Z values together to define the location of each vertex. This method does not share values. This difference is most easily seen in an example of a particle:

Particle from sides method:

Swarm[i] =[x1,x2,x3,z1,z2,z3,rho]

Particle from points method:

Swarm[i] = [[x1,z1],[x2,z2],[x3,z3],[x4,z4],[x5,z5],[rho]]

Since there is no sharing of values, the points method is slightly more complex, but it is much more robust as a result, allowing each point to move independently of the others, thus generating more complex objects. Integrating the points method required vast changes to the infrastructure of the software due to the difference in shape of the two swarms. This revamping was constructive however, leading to a new manner of initializing the swarm, and a new technique for detecting infeasible models. Previously, initializing the swarm was a simple matter of randomly picking X,Z, and rho values from a range of acceptable values. For the points method, however, this could lead to initial models with unacceptable shapes. Instead, the new version first divides 360° by the

number of sides of the object to get an even increment. Then using a combination of random depth to the center of the object, random radius to each point and the angle increment, the points are oriented in a circular manner. This method allows the objects to start in an acceptable shape 100% of the time. An example of what an initial swarm could look like is shown in Figure 5.2.

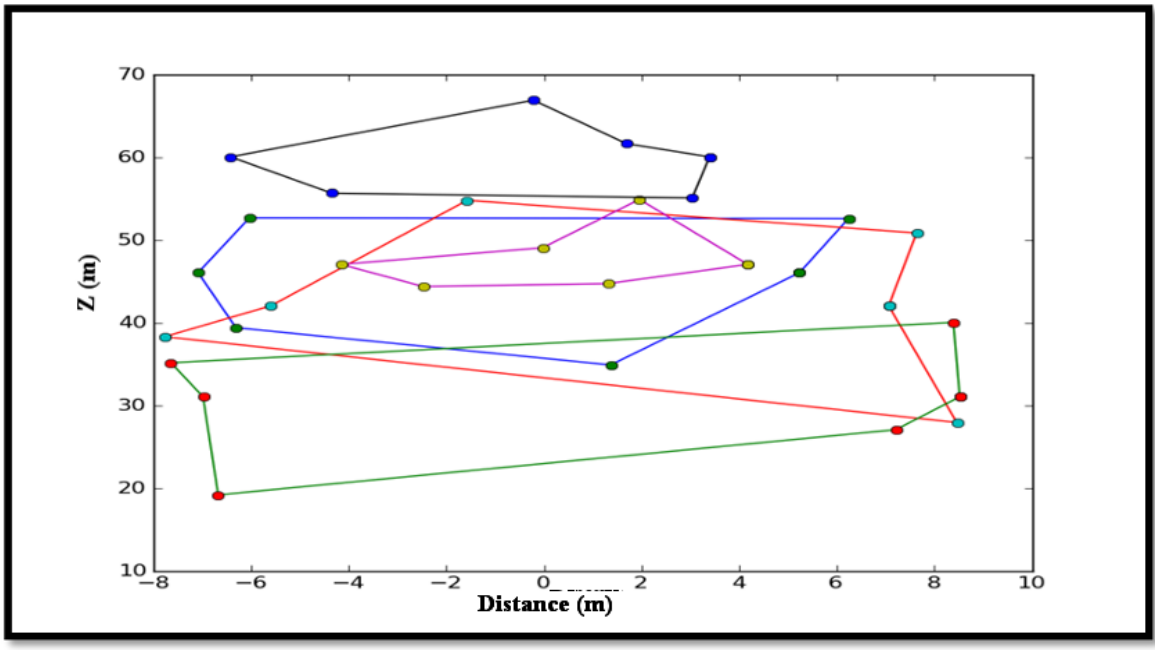

Figure 5.2: Plot of five possible shapes created by the new initialization procedure used for the points method.

In previous versions of the inversion algorithm the process of checking for infeasible models was needlessly complex for models with more than four sides. The process was basically a piece by piece check, which evaluated the location of every other point with respect to the current test point. If for example the point being considered was

the bottom left corner, the position of every other point would need to be evaluated to ascertain if it is indeed the bottom left corner. This process is also the reason that the corner switching method was put aside. The new method however does not rely on this tedious process, and is composed of a primary and a secondary check. A feasible model, in this thesis, is actually a simple polygon, and an infeasible model is a complex polygon (see Figure 5.3).

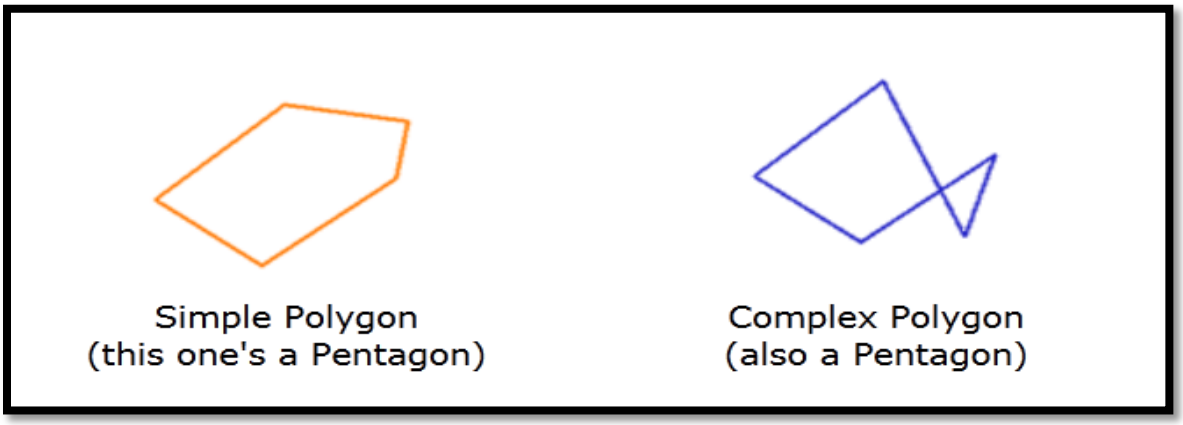

Figure 5.3: Depiction of simple and complex polygons: simple represents a feasible model and complex a non-feasible model (Pierce 2013).

The primary checking procedure involves calculating the exterior angles of the polygon: the sum of the exterior angles will always add to 360° in a simple polygon, no matter how many sides it has (Pierce, 2013; Figure 5.4), but will not in complex polygons.

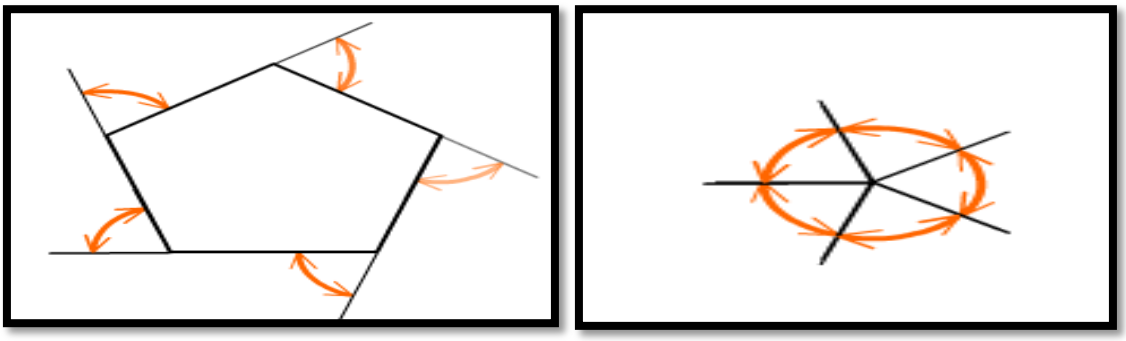

Figure 5.4: Demonstration of how the exterior angles of a polygon add to 360 degrees (Pierce, 2013).

There are however some relevant exceptions where the exterior angles of complex polygons will also add to 360°. These polygons will always have at least two sides crossing, as seen in Figure 5.5, and for this reason a secondary check is applied to models which have exterior angles adding to 360°.

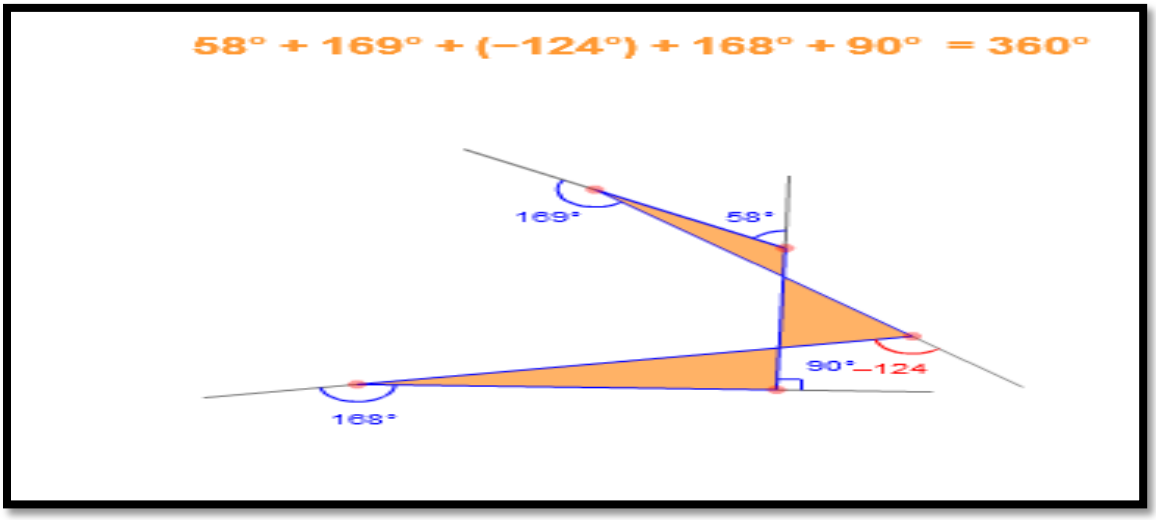

Figure 5.5: Example of a complex polygon with exterior angles adding to 360 degrees (John, 2013).

The secondary checking procedure is designed to detect crossing lines. It does this by calculating vectors between each pair of points, and evaluating if those vectors cross. The combination of these two methods is much more robust than those previously used (Section 4.2.4), and is far simpler to implement. It also allows for the use of additional sides without changing the penalty detection methods. Another advantage of this method, when used in tandem with the points method, is that the points are no longer tethered to specific corners, and are free to rotate during the inversion process without causing penalties to occur.

The five-sided model remains unchanged from Chapter 4 as an inversion target, and will be used for testing the newly developed constraint handling methods described in Section 5.3. An additional six-sided inversion target is introduced here, which is now possible due to the use of the points method of parameterization. The six-sided model has several characteristics in common with previous inversion targets. Similar to the four and five-sided models it extends from X = -10m to X = 10m. This model is a symmetric polygon, which is located between the depths of 21.34 and 38.66m. The density value is also the same, 1000.0 kg/m^3. Figure 5.6 is a diagram showing the six-sided model, and the synthetic data of the four, five and six-sided models. The gravity curve of the six-sided model is represented in dark blue, and is extremely similar in shape to the gravity curve of the four-sided model (light blue). Both the four and six-sided models are much broader than the synthetic data of the five-sided model (red).

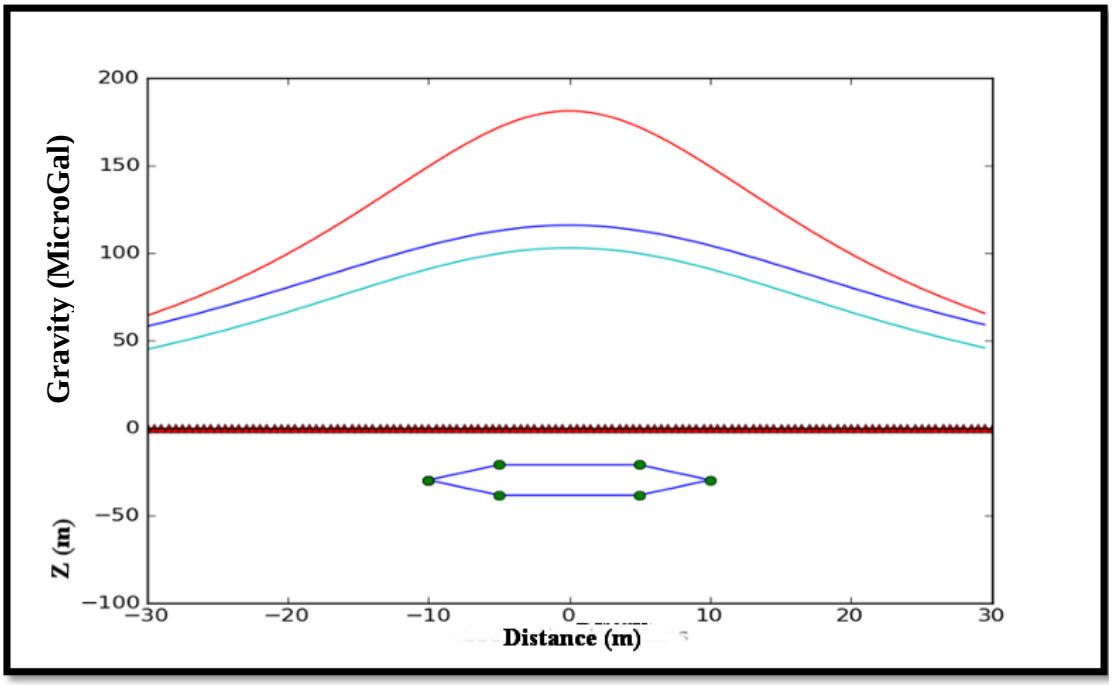

Figure 5.6: Depiction of the six-sided model and synthetic data from the 4,5 and 6 sided models, represented in light blue, red and dark blue respectively.

5.3 Constraint handling methods and required changes.

As described in previous sections, the corner switching method was the first attempt at constraining the algorithm to deal with infeasible solutions, i.e., shapes that are turned inside out. This method was a quick and simple way to fix the problem, however there were some issues with that method which were discovered early on. When the five-sided model was introduced the corner switching method was found to be overly complex, and the complexity would only increase with the number of sides. The next attempt at a constraint handling method was to use penalty functions, which are the traditional means for solving constrained non-linear optimization problems (CNOPs). When using this

approach the constraints are effectively removed and a penalty is added to the objective function value (in a minimization problem) when a violation occurs.

$$\boldsymbol{Objective\ function} = \sum_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{d_i^{obs}} - \mathbf{d_i^{calc}})^{\mathbf{2}} + \boldsymbol{Penalty\ value}$$

Hence, the optimization problem becomes one of minimizing the objective function and the penalty at once. Penalty functions can be stationary or non-stationary. Stationary penalty functions add a fixed penalty when a violation occurs, whereas non-stationary penalty functions add a penalty proportional to the amount the constraint is violated. Another factor to consider with this method is optimizing the penalty values used. They need to maintain a balance between obtaining feasibility while also finding optimality: a value which is too small may cause the problem to never converge, while a value which is too large may cause early convergence to a non-optimal solution.

The third method of constraint handling experimented with here was the Rejection of Infeasible solutions method (RIS method). The RIS method differs from the penalty function method in that it discards any violations detected instead of adding a penalty. The drawback of this method is that for some types of problems there may be some infeasible solutions that are better than feasible ones. This however does not generally apply when the method is used to invert geophysical data, since any infeasible solution would be either turned inside out, or outside of the survey region or above the Earths surface. When adapting this technique into the global version of PSO, the initialization process involves forcing all of the particles into feasible space before any evolution of the objective function can begin (Hu et al. 2003). During evolution only those particles that

remain in the feasible space can be considered for new values of *pbest* and *gbest*, while the others are rejected. The idea here is to accelerate the iterative process of tracking feasible solutions by forcing the search to contain only solutions that do not violate any constraints. Another obvious drawback of this method is that for CNOPs with extremely small feasible space the initialization process can become impractically long. Once again this is not an issue in this case due to the manner in which the particles are initialized for this algorithm.

5.4 Results for five-side inversions with penalties, and six-sided points method.

5.4.1 Inversions of five-sided models with a penalty function value of 500

This section will discuss the results obtained from the tests performed on five-sided models, using the sides method of parameterization, with both the penalty function and RIS methods of constraint handling. Results from tests performed on the six-sided model using the points method of parameterization will also be discussed in this section. The testing using penalty functions involved the use of two different penalty values. The first was a small penalty of 500, and the second a much larger penalty of 50,000. This was done to either find penalty values which performed well or determine which of these would cause the algorithm to fail. Each set of tests using the sides method were performed over five thousand iterations, and repeated twenty times. The recorded values were the misfit, the time taken, the x and z values, and the density.

Section 5.3 stated that using a penalty function that is too small may cause the inversion to never converge. The first value tested was a penalty value of 500, which was

added to the misfit function every time a violation occurred. Table 5.1 shows the results of this test suite, and it is clear that this penalty value is far from optimal. The average objective function achieved amongst this suite of 20 inversions was 598.25, with an average time to completion of 21.1 minutes using Cython file type. During this set, zero out of twenty achieved an objective function value small enough to trigger the stop parameter. Also evident from Table 5.1 is the number of tests which were unable to find feasible models in the allotted time; these are highlighted in yellow, and account for nearly half of all the inversions done. Disregarding the infeasible solutions, the highest objective function returned was 2951.25, followed by 2083.02. The lowest objective function achieved was 0.65, leading to an average objective function of 608.0, which is actually slightly higher than the overall average. Figure 5.7 is a plot of the feasible solutions objective function values, and the average objective function of those solutions. It can be seen that the majority of the values fall below the average line, with three values lingering near it, and two are far above the line. Using the same criteria established in Chapter 4, the values which fall below the average line would be considered quite good, the values near the average may be acceptable and the ones far above the line are most likely poor results.

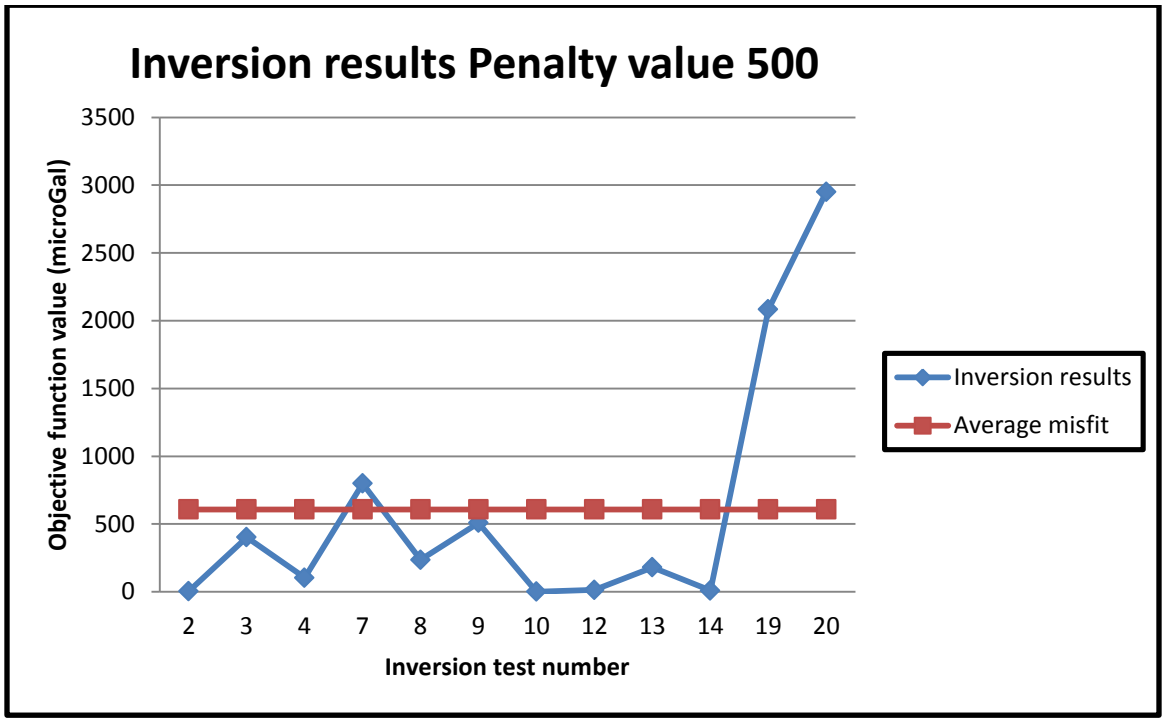

Figure 5.7: Plot of inversion results and average objective function value from constrained five-sided tests using a penalty function with a value of 500.

Table 5.1: Inversion results for constrained five-sided example using a penalty function value of 500.

Test #	objective function	Time Taken(min)	X1	X2	X3	Z1	Z2	Z3	Rho
1	541.78	22.65	-23.66	1.18	14.07	57.6	7.87	39.12	1199.58
2	5.04	23.925	-15.07	0.33	15.13	26.92	10.29	26.92	1155.255
3	402.85	22.5	-3.67	-2.035	7.25	34.25	6.41	34.33	2058.52
4	104.15	23.565	-8.51	-8.33	8.55	15.3	15.3	27.37	1430.28
5	505.99	21.195	-14.99	0.39	12.12	46.3	10.75	34.05	1982.8
6	539.35	22.42	-25.6	0.022	25.07	81.64	6.42	48.83	979.97
7	798.81	20.67	-3.54	14.94	2.4	11.45	8.71	44.72	1582.06
8	235.66	21.815	-7.66	3.53	4.75	24.28	9.05	31	1706.42
9	509.3	20.92	-6.88	28.84	6.88	13.82	13.82	31.6	1232.488
10	0.659	20.48	-13.55	-0.057	13.65	26.59	10.49	27.32	1246.69
11	590.39	21.545	-28.48	3.21	16.03	37.11	10.24	29.7	1093.63

12	14.33	21.56	-10.81	1.05	10.19	21.95	10.01	29.08	1088.71
13	181.2	19.925	-6.93	-2.28	8.366	21.77	7.78	33.58	1069.2
14	9.99	19.945	-7.83	2.15	7.46	16.19	11.92	31.28	1144.68
15	574.21	20.26	-2.74	-19.81	4.21	12.51	9.53	39.31	1593.54
16	765.36	20.41	-17.46	0.51	6.57	98	4.58	55.88	1969.89
17	513.83	20.785	-20.78	2.47	9.2	41.65	10.18	32.32	1564.9
18	637.62	20.875	-5.966	-5.83	23.19	29.07	15	25.8	2680.4
19	2083.02	19.8995	-9.75	2.89	3.08	37.72	3	37.73	1424.07
20	2951.44	19.22	-1.78	1.882	1.884	7.43	7.43	68.46	1719.77
average	598.249	21.22823	-11.782	1.2526	10.0025	33.0775	9.439	36.42	1496.143
Feasible	608.0374	21.20							

Table 5.1 continued

Test numbers 10, 2, and 14 were the best results returned from this suite of inversions, with misfit values of 0.65, 5.04, and 9.99 respectively. These correspond to excellent results as seen in Figure 5.8.

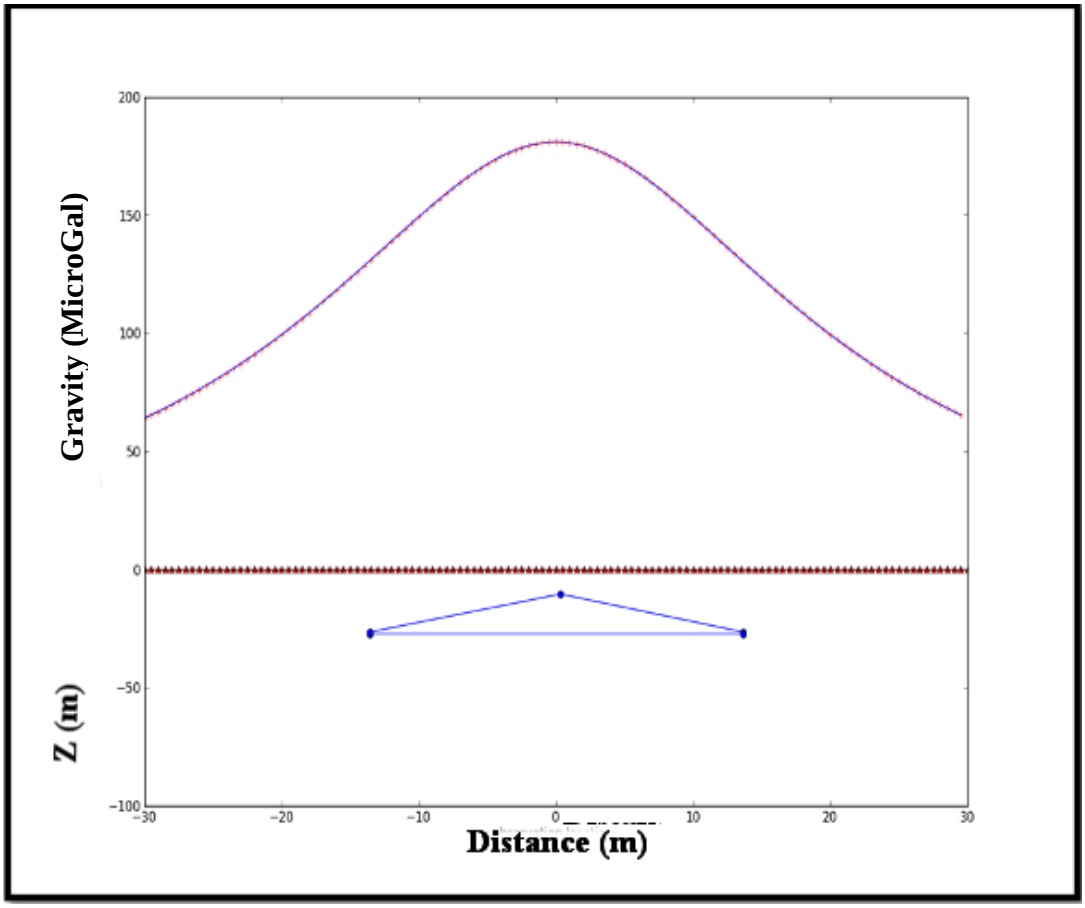

Figure 5.8: Plot of the model returned in test inversion 10, with predicted and synthetic data indicated by red crosses and the blue line respectively.

Figure 5.8 shows that the predicted data (red +) fall on or near the synthetic data (blue) over the whole extent of the curve. The model shape is basically a triangle, with the center point located at the position X = -0.057m, Z = 10.49m which is extremely close to

the position of the center in the target model X = 0.0, Z = 10.0. The object extends from X = -13.55m to X = 13.65m, which is also very close to the target model. It is compensating the smaller size with increased density, 1246.69 kg/m^3 up from 1000.0 kg/m^3. The model appears to only have three vertices, however there are indeed five. The left and right sides are contracted together making it appear as if there are three vertices. Results closer to the average misfit value include those for runs 3, 8, 9 and 7 with objective function values of 402.85, 235.66, 509.3 and 798.81 respectively. Figure 5.9 shows the models returned from test inversions 3 and 8 (green and red respectively) as well as the predicted data from those models (matching colors), overlapping the synthetic data from the target model (blue curve). Similar to previous examples the shape and the placement of the predicted data curves are very similar to the synthetic data curve. Model 3 has its center point located to the left of the origin, causing a corresponding shift in the predicted data curve. Opposing this is model 8, with a center point position on the right on the origin, causing a rightward shift in the predicted data. Overall both models produce very acceptable sets of predicted data.

Test models 7 and 9 also lie close to the average objective function value in Figure 5.7. Plots of the predicted data from these models are shown in Figures 5.10 and 11. It can be seen that these two sets of predicted data are actually not representative of their misfit values, especially model 9. Figure 5.10 seems to be a very reasonable result, matching the shape and amplitude of the synthetic data well. The objective function value of this model is 798.81, but comparing the two curves it seems like it should have a much lower misfit. This theory is strengthened by Figure 5.11, whose objective function value

is 509.3, but the predicted data is more akin to a misfit of less than 10. This uncertainty can be easily explained by viewing the actual shapes of the models, as seen in Figure 5.12. The center point of both models is located far to the right of the main body, causing them to be mistakenly assessed a penalty when they are both feasible shapes (model 9 appears to be pinched out due to scale, but the value of Z2 is actually slightly higher than Z1). This was caused in part by an error in coding which adds penalties to models with center values outside of the model shape, but also because the penalty was not strong enough to guide the models towards better shapes, which will be confirmed in the results of tests described below.

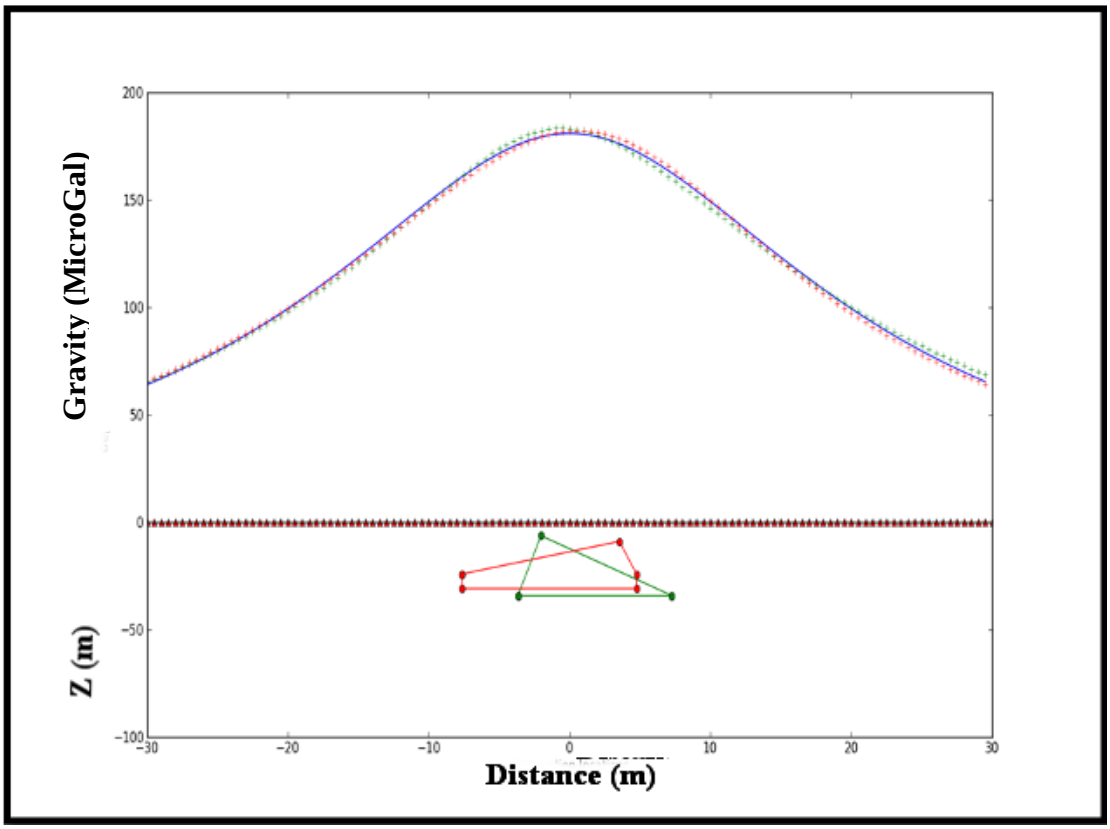

Figure 5.9: Plot of the models returned in test inversions 3 and 8 (green and red dots and lines respectively), with predicted and synthetic data (green and red crosses, and blue line respectively).

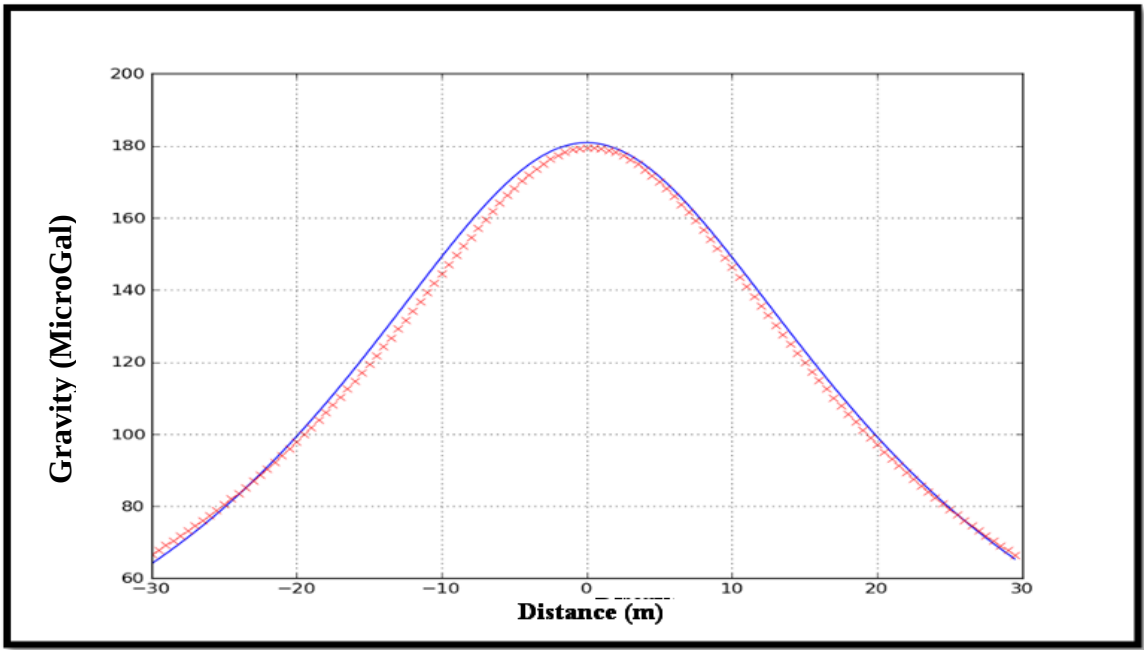

Figure 5.10: Predicted data (red+) and synthetic data (blue line) for run 7 of five-sided example constrained with a penalty value of 500.

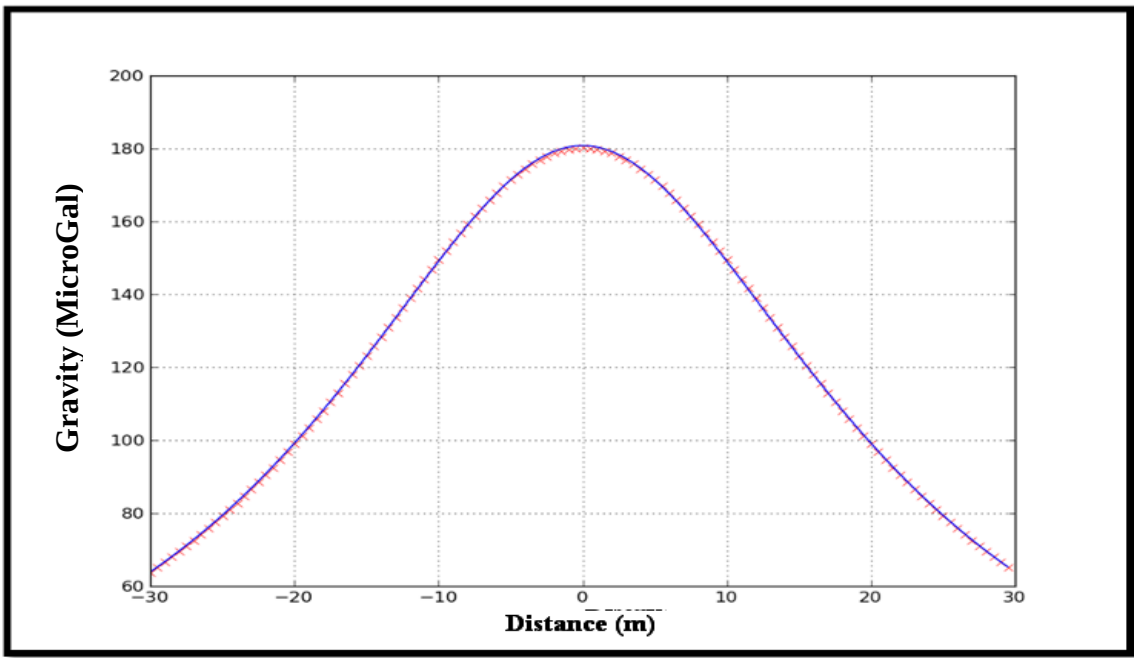

Figure 5.11: Predicted data (red+) with synthetic data (blue line) for run 9 of five-sided example constrained with a penalty value of 500.

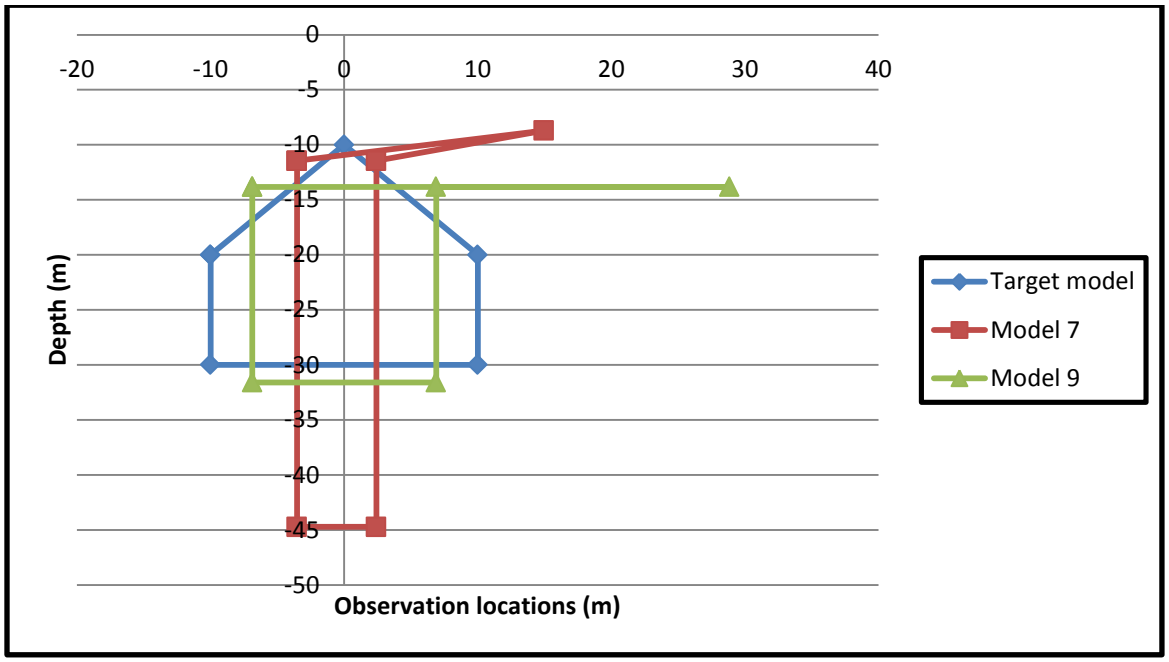

Figure 5.12: Plot of model shapes returned from test inversions 7 and 9 (and the true model) for the five-sided examples constrained using a penalty function value of 500.

Test inversions 19 and 20 returned very poor results, having objective function values of 2083.02 and 2951.44 respectively. Figures 5.13 and 14 show the predicted data results for these models, and it can be seen that they do not match up well. The shapes are too narrow, and the maximum amplitudes are too high, nearly 10 microGal of difference in both cases.

Overall the results from inversions using a penalty function with a stationary value of 500 were unsuccessful. Although 12 of the 20 inversions were successful in obtaining feasible models, two of these were very poor results, leaving the remaining 10 with results ranging from excellent to merely acceptable. The fact remains however that nearly

half of the inversion results were infeasible solutions, which inevitably was the deciding factor in deeming this penalty value a failure.

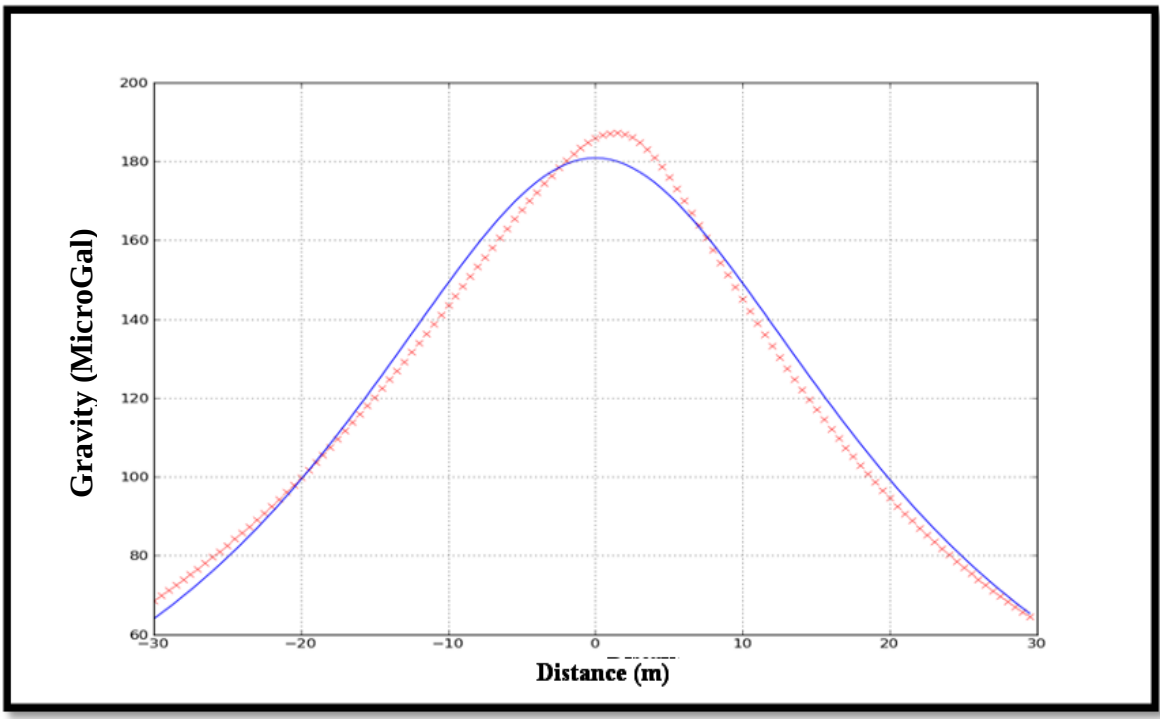

Figure 5.13: Predicted data (red+) and synthetic data (blue line) for run 19 of five-sided example constrained with a penalty value of 500.

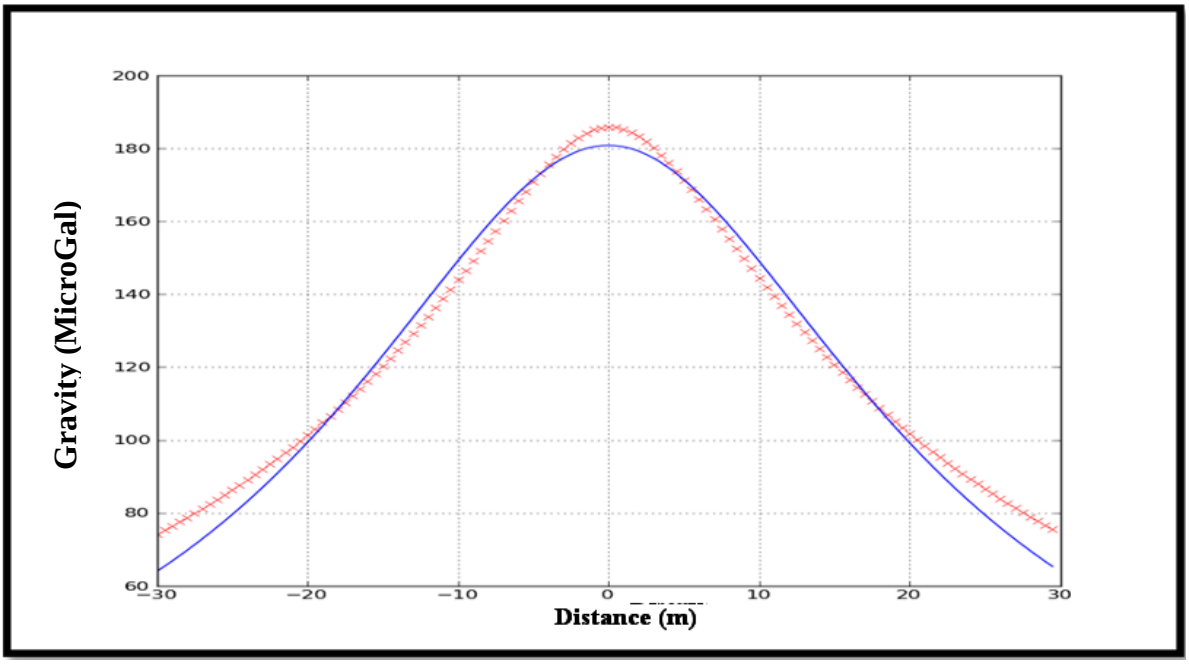

Figure 5.14: Predicted data (red+) and synthetic data (blue line) for run 20 of five-sided example constrained with a penalty value of 500.

5.4.2 Inversions of five-sided models with a penalty function value of 50,000

Increasing the penalty value from 500 to 50,000 drastically improves the results returned, as can be seen in Table 5.2. During this test suite an average objective function value of 1452 was achieved, however this is a misrepresentation of the results. A single extreme outlier is present which clouds the true quality of the data, as seen in Figure 5.15. Disregarding this value gives a more informative representation of this data set. The average objective function value becomes 54.57, with a new high value of 141.65, and two equal low values of 0.1. The average time to completion for these inversions was

19.07 minutes, using Cython scripts. Two of the twenty inversions had objective function values low enough to trigger the stop parameter, both contributing to an average number of iterations of 390. Increasing the penalty value to 50,000 has also eliminated all cases of infeasible results in this set, which is a vast improvement over the previous suite, which had eight such cases.

Since it has already been established that models with misfit values as high as 607 are deemed successful inversions, the majority of the results obtained from this set need not be shown since they are all less than 141, as seen in Figure 5.16. Results from model 20 and the outlier run 16, with objective function values of 141 and 28013.1 respectively, will be presented, since they had the highest objective function values. Figure 5.17 shows the final model returned in inversion test number 20, as well as the gravity curve created from that model. It is clear from this figure that this model produces more than an adequate match for the synthetic data, and since this was the worst out of the 19 inversions, it verifies the quality of the other results.

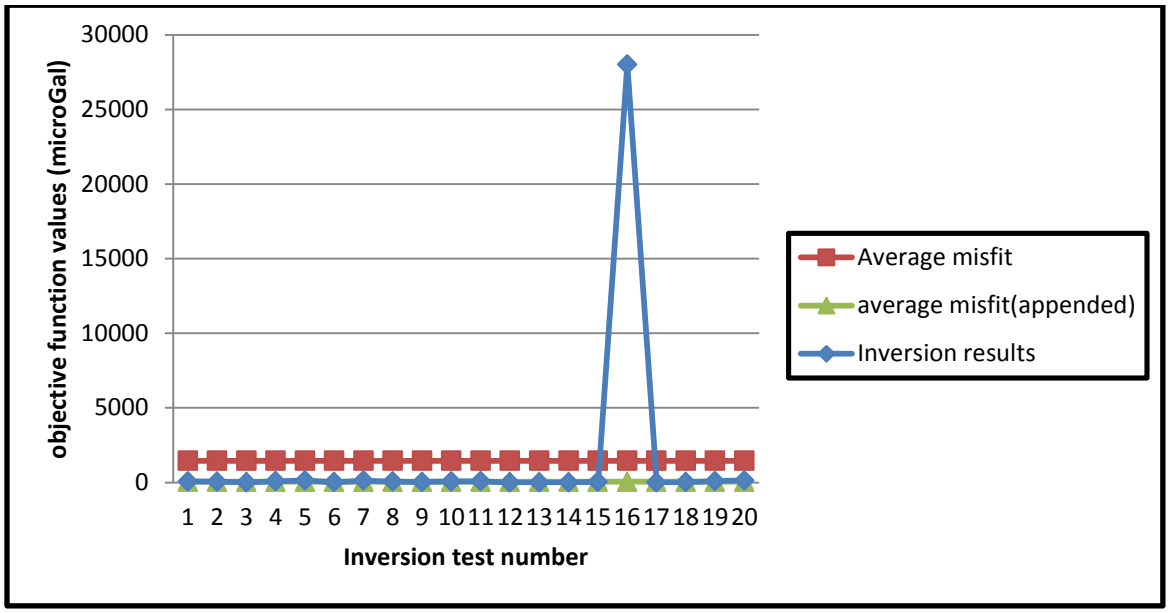

Figure 5.15: Line plot of objective function values obtained from inversions using a penalty value of 50,000.

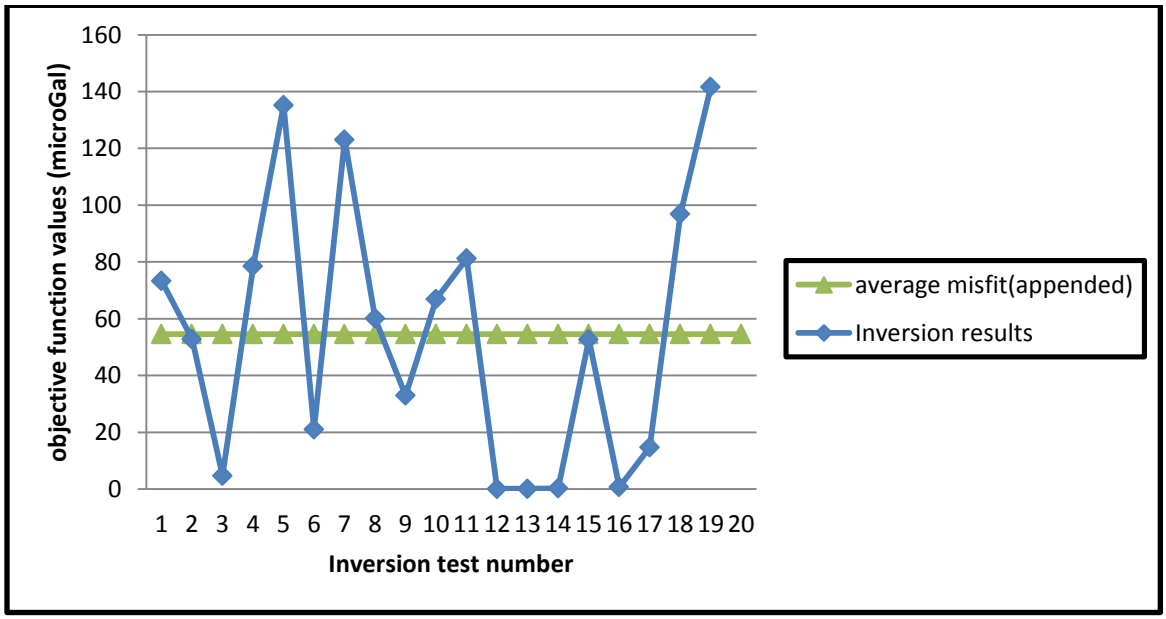

Figure 5.16: Objective function value omitting the outlier of 28,013

Table 5.2: Results obtained from the five-sided constrained inversions with sides method using a penalty value of 50,000.

Test #	objective function value	Time Taken(min)	X1	X2	X3	Z1	Z2	Z3	Rho	convergence iteration
1	73.28	17.505	-5.5	0.41	5.91	30.24	9.22	31.23	2352.9	
2	52.84	17.53	-4.14	0.66	4.13	12.97	12.97	36	1631.21	
3	4.7	18.195	-14.36	0.16	14.33	16.44	6.42	32.06	501.44	
4	78.48	21.125	-6.69	-3.92	7.79	19.08	11.81	29.59	1462.78	
5	135.17	21.555	-15.34	2.63	11.9	23.46	15.2	23.46	2598	
6	21.07	21.97	-10.14	-1.35	11.89	26.65	12.02	26.92	1785.1	
7	123.08	22	-13.82	-0.89	15.06	22.84	15.14	22.97	2523.32	
8	60.12	21.87	-27.44	0.56	26.39	22.74	5.7	23.64	552.92	
9	32.93	21.85	-7.11	7	7	11.85	11.19	36.5	872.74	
10	66.9	22.075	-20.68	-0.49	21.17	18.92	5.59	25.76	500	

11	81.18	22.07	-11.99	2.56	9.66	23.45	13.32	25.26	1976.47	
12	0.1	1.9	-9.54	-0.008	9.55	25.97	11.55	27.75	1749.97	433
13	0.1	1.57	-12.5	0.007	12.5	25.81	10.38	27.69	1241.17	347
14	0.277	18.795	-9.99	-0.12	10.06	22.53	10.65	28.59	1244.16	
15	52.73	21.82	-4.52	-2.69	5.66	21.17	11.31	30.86	2049.395	
16	28013.1	21.89	-1.36	-1.36	4.36	40.75	30	75.21	1529.57	
17	0.725	21.395	-14.88	0.00067	14.88	26.9	9.85	27.44	1100.98	
18	14.66	21.325	-4.54	-0.0067	4.61	14.82	13.35	32.92	1774.66	
19	96.83	23.955	-13.2	0.029	13.15	22.12	14.14	23.71	1978.86	
20	141.65	23.94	-4.7	-2.22	4.63	11.48	11.48	39.55	1206.022	
average	1452.496	19.21675	-10.622	0.048099	10.7315	22.0095	12.0645	31.3555	1531.583	390
average(no 16)	54.56958	19.07605	-11.109	0.122209	11.06684	21.02316	11.12053	29.04737	1531.689	

Table 5.2 continued

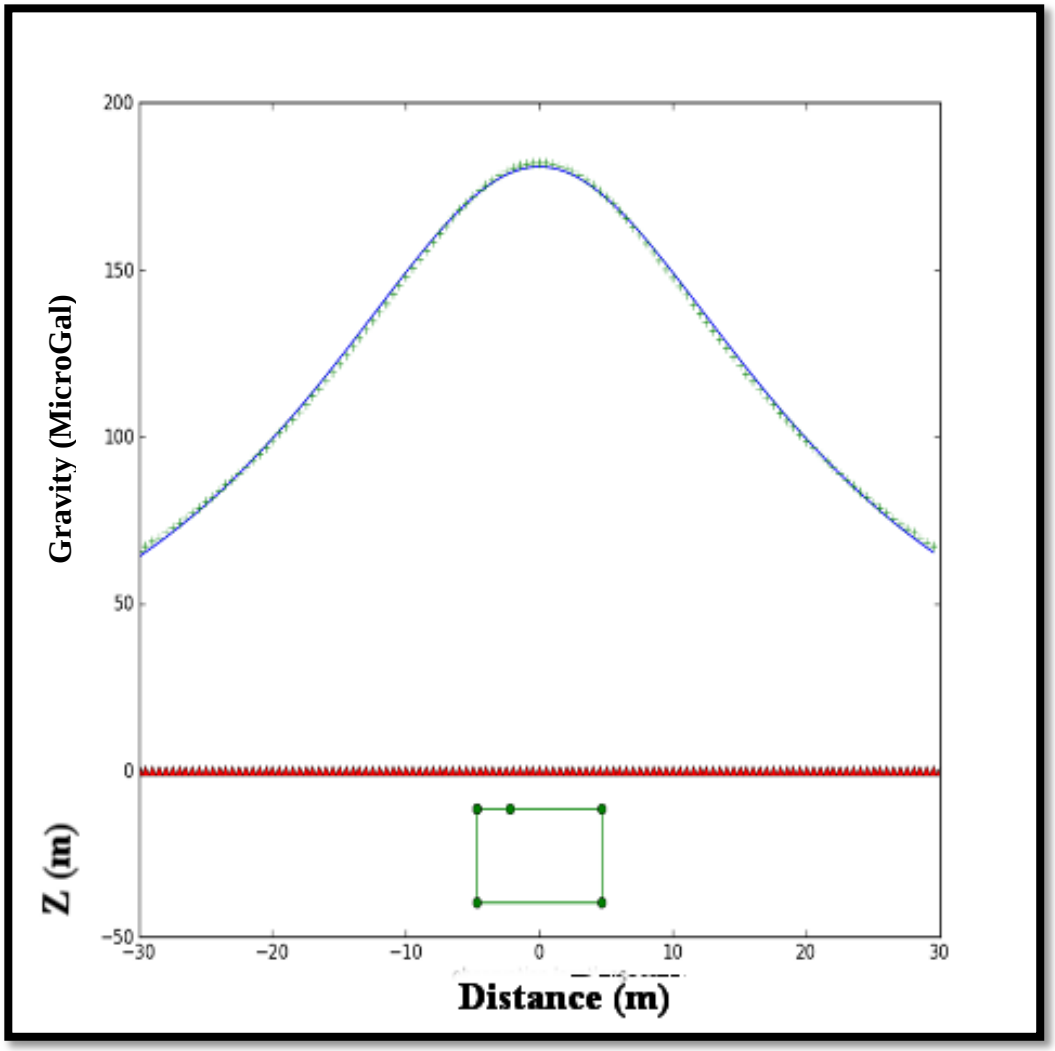

Figure 5.17: Predicted data (red crosses) and model for run 20 using penalty value 50,000.

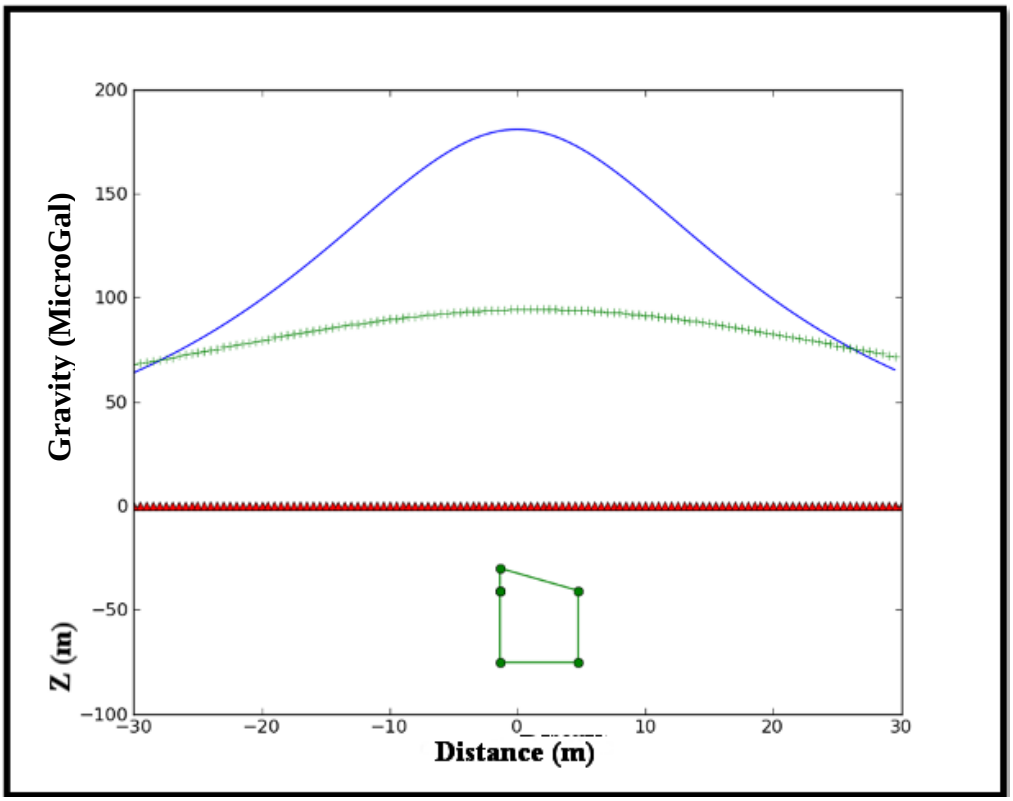

Figure 5.18: Predicted data (red crosses) and model for run 16 using a penalty of 50,000.

Figure 5.18 shows the resulting model from inversion number 16, as well as the predicted and synthetic gravity curves. Unlike previous results, this set of data is an extremely poor match in shape and maximum amplitude. At its peak value it is nearly half the amplitude of the synthetic data. The cause of this failure is believed to be the same coding error mentioned in Section 5.3.1 which affected models 7 and 9. The center point of the model appears to be attempting to move farther to the left, but since the penalty value is now greater than the current misfit value, the algorithm wouldn't permit the point to travel in that direction. If the point where to move further to the left, the misfit would increase by 50,000 due to the error in the penalty function. This must have caused the other particles

to gather at that same area, causing the velocities to drop to zero, halting any further progress of the PSO procedure.

The results obtained from inversions using a penalty value of 50,000 were excellent overall. Excluding a single outlier, caused by an error in coding, the average misfit value for this suite was 54.65, which was the lowest of any test suite obtained for the five-sided model thus far. This, along with the complete lack of infeasible solutions returned, confirms that this method performs extremely well with PSO when an appropriate penalty value is used. The RIS method of constraint handling will be discussed in the next section. Results from the RIS will then be compared with the results obtained in this section in order to determine which of the two methods is the most promising, and therefore which will be adopted in the points method inversion algorithm.

5.4.3 Inversions of five-sided models with RIS method.

The results of the tests performed with the rejection of infeasible solutions (RIS) method of constraint handling were less than favorable. The average overall misfit achieved was 23460, with a high misfit value of 68782, and a low misfit value of 10.66. Similar to previous test suites, several distinct divisions of quality are observed in this set. The divisions fall into three subsets, the first subset had five models ranging in misfit from 10.66 to 594.22. From previous examples these will most likely be considered successful inversions. The second subset also has five models, which are possible successes, with misfit values ranging from 986.83 to 1792.53. The final subset contains the remaining ten inversions which are definite failures, with misfit values ranging from 13912 to 68782.

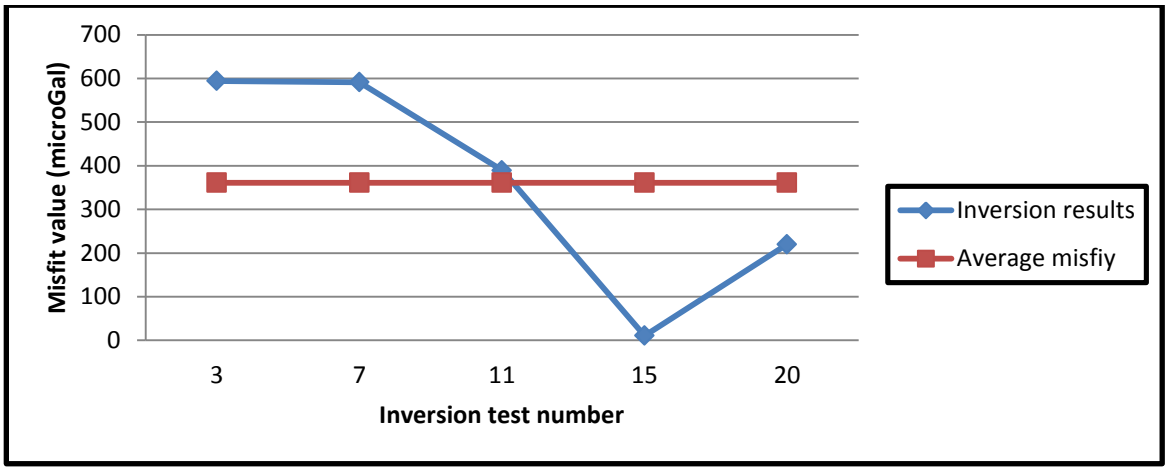

Figure 5.19: Set 1 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations.

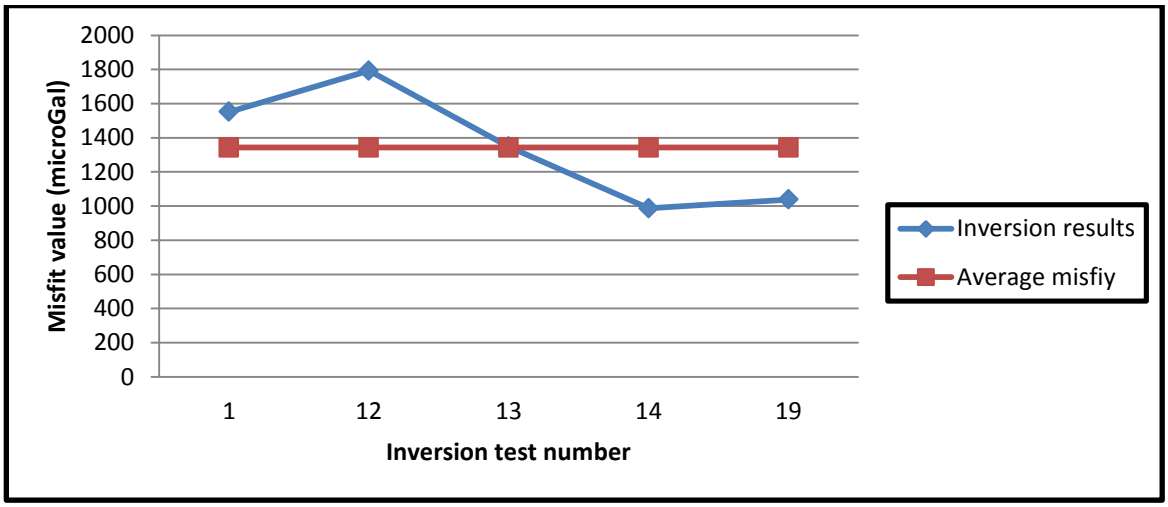

Figure 5.20: Set 2 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations.

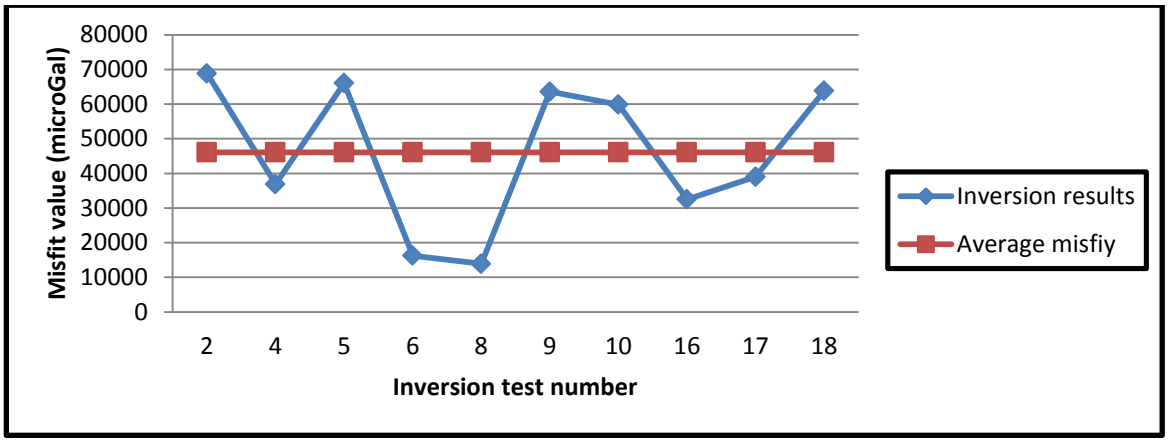

Figure 5.21: Set 3 of obtained misfit and average values from RIS method experiments for five-sided objects with 5000 iterations.

Figures 5.19,5.20 and 5.21 show the inversion results of each subset along with the average of each subset. Figure 5.19 shows the misfit values do not exceed 600, and the average misfit is just under 400, at 361.1. Figure 5.20 depicts the results of the second subset, which has an average misfit of 1343.3. Figure 5.21 shows the result for the third subset, which has an average misfit of 46069. Drawing on previous conclusions and Figure 5.22, a misfit plot for model 7, it is clear that this model would be considered a successful inversion. Since model 7 has a misfit which is extremely close to the highest value in this set, and all other values in this set fall below it, it is safe to assume that the other models in this set are also successful inversions. This fact is further corroborated by Figure 5.25, which shows data difference plots for models 7, 14, 19 and 12. It can be seen that the difference between the predicted data of model 7 and the synthetic data has a maximum amplitude of approximately 4 microGal, while the majority of the difference values are actually located between plus and minus 2 microGal, which is indeed an

acceptable difference range. The additional models were selected to investigate possible success or failure of subset 2 since subset 3 contains all failures. Model 14 has the lowest misfit in subset 2, and model 12 has the highest misfit of subset 2; determining the outcome of these inversions will help define the success or failure of this subset. Figure 5.23 shows the predicted data for model 14 from subset 2, with a misfit value of 986. While considering Figure 5.23 it is necessary to point out that the shape of the predicted data is very similar to previously examined results. The left and right end points are higher than the synthetic data, similar to the peak value, which is also higher and to the right of the synthetic curve, while other sections fall close to the curve. If this was the sole avenue of exploration this model could be considered a success, however there are other options to consider. Figure 5.25 also contains data difference information for model 14. It can be seen that the amplitude of the difference is constantly larger than that of model 7. In this example, unlike for model 7, the majority of the differences lie in the region of 2 microGal or greater with a maximum amplitude of nearly 6 microGal, which leads to the decision that it is actually a failed inversion.

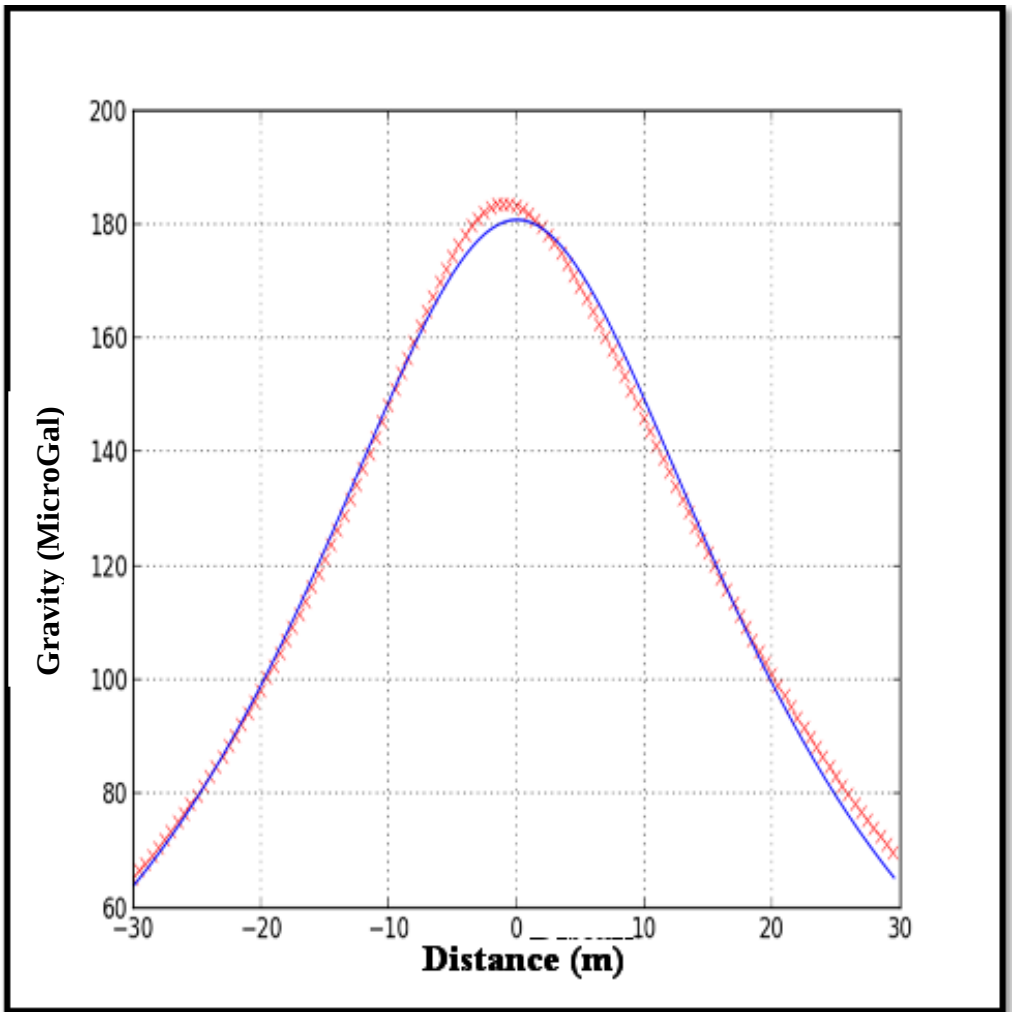

Figure 5.22: Plot of predicted data of model 7 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 7 is 591.

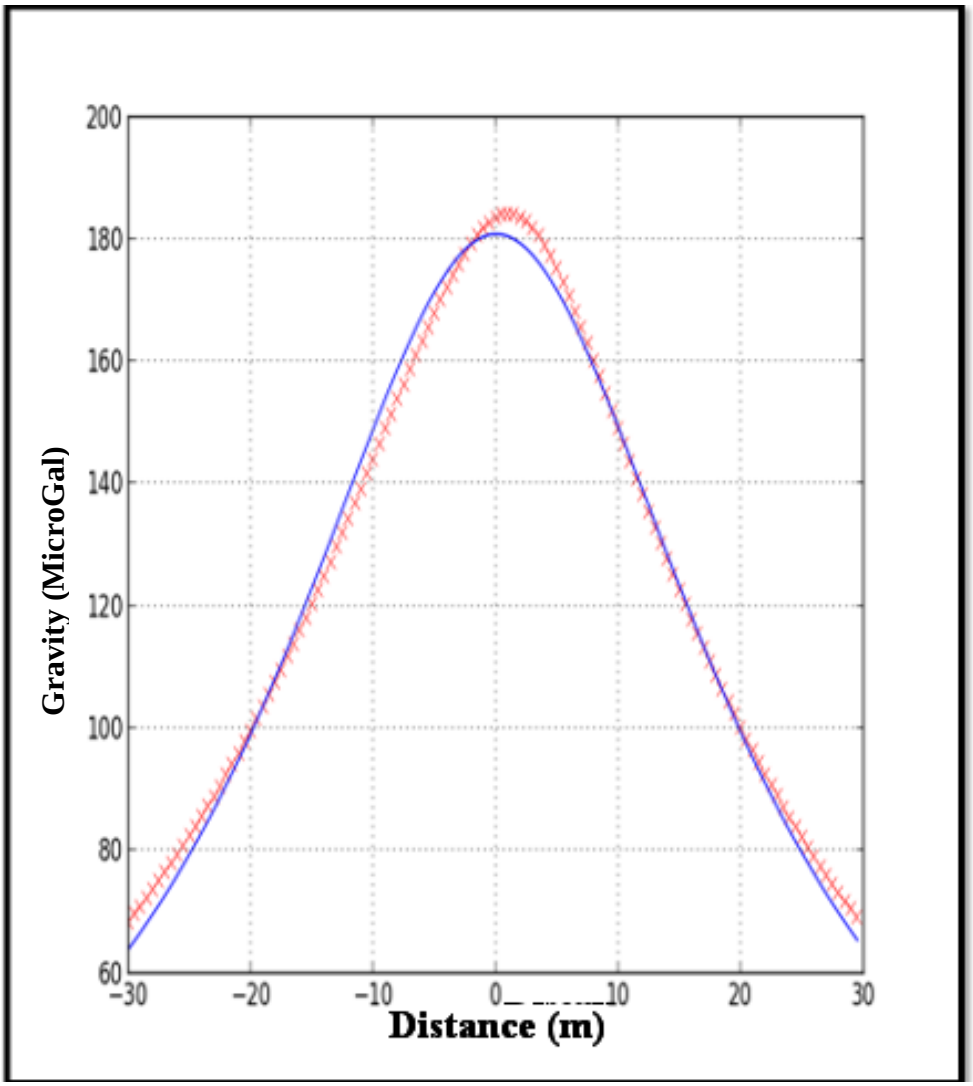

Figure 5.23: Plot of predicted data of model 14 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 14 is 986.

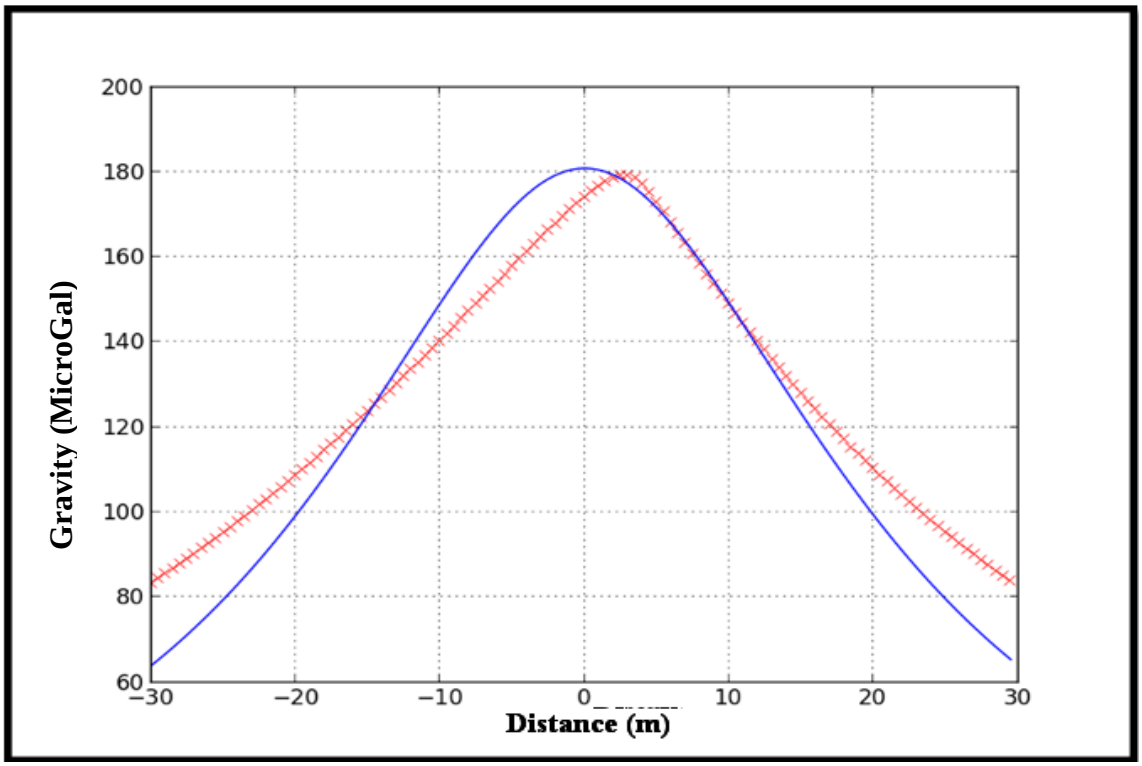

Figure 5.24: Plot of predicted data of model 8 (red) of the RIS constrained tests with synthetic data (blue). The misfit value of model 8 is 13912.

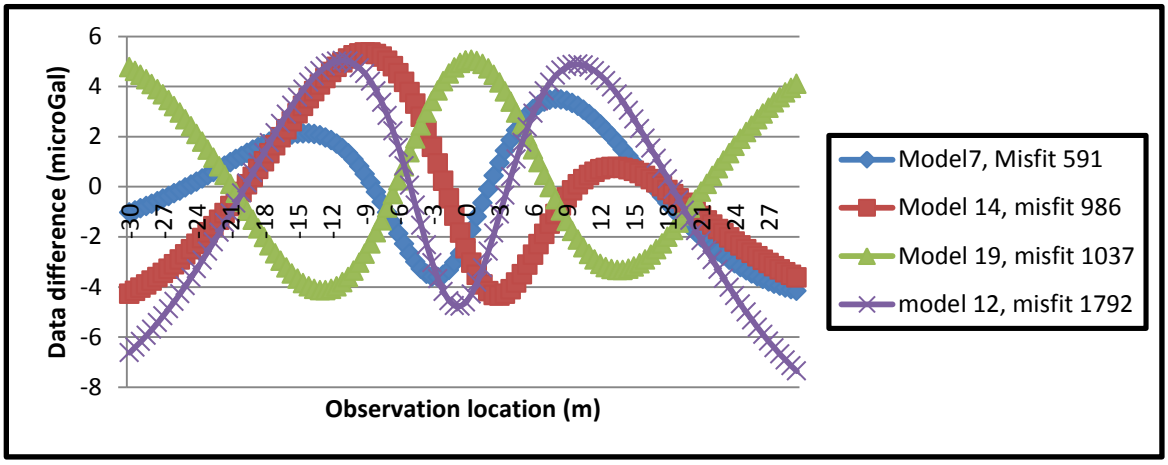

Figure 5.25: Data difference plot for predicted data for models 7, 14, 19, and 12 from the RIS, five-sided constrained inversions.

Since model 14 has the lowest misfit value of this set, it is only logical that the remaining models of this set, including subset 3, are considered failed inversions. This is supported by Figure 5.25, which also shows data difference information for models 19 and 12. These examples are comparable to the information derived from model 14. This is demonstrated further in Figure 5.24, which shows the misfit plot of model 8, the best result returned in subset 3. The predicted data here is too narrow, and is centered to the right of the synthetic data, leading to a mismatch in shape, and therefore a large misfit value. Unlike previous results where failed inversions came from infeasible solutions, or coding errors, it does not appear to be the case here. The results from this section were all feasible solutions, which can be seen in Figure 5.26, a plot of the lowest models returned in each subset as well as the base model. The figure shows that these are indeed feasible solutions, and it does not appear that the center point of any model is probing the sides, causing a failed inversion by coding error.

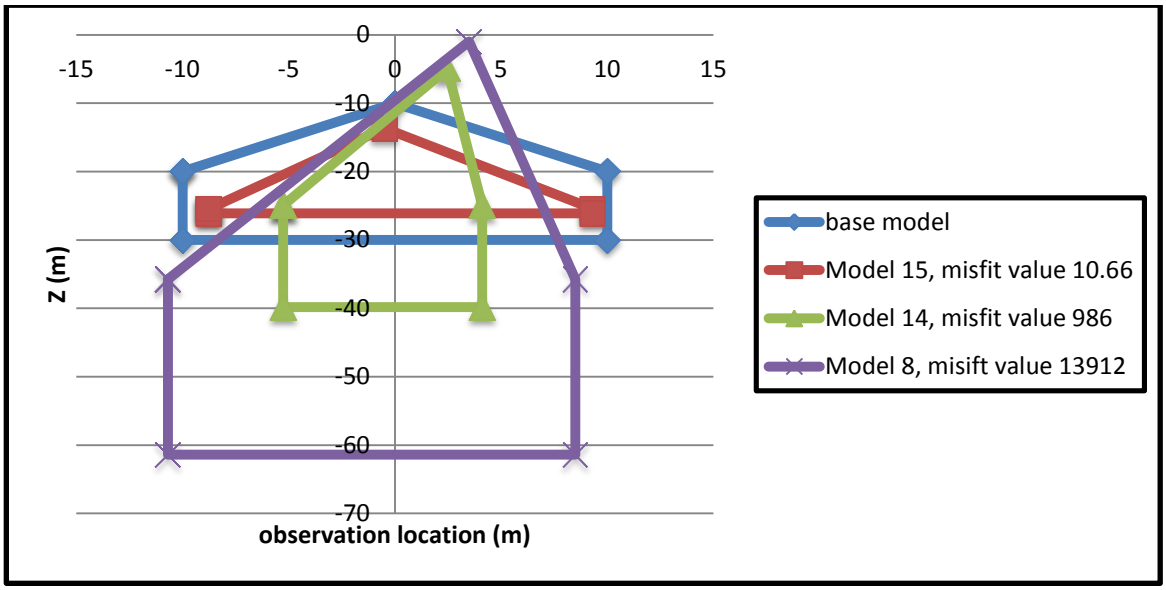

Figure 5.26: Plot of constrained models for each of the three subsets of results.

Given that the cause of these failures is not infeasibility or coding errors, it is believed that a greater number of iterations could allow the inversions to find better solutions. To this end, a small experiment of ten inversions was carried out using the same parameters as previously applied with an increased number of iterations, namely, 10,000.

Figure 5.27 and Table 5.3 show the results obtained from the additional experiments of the RIS method. It can be seen that there are two large outlier values in this set, one with a misfit value of 89639, and the other with a misfit value of 9025. The average overall misfit achieved in this set was 10208, however with the two outlier values removed the data becomes much better, with an average misfit of 427.21. The remaining eight inversion results appear to show great improvement over the previous suite. Using previous conclusions, five of the inversions would be considered successful; one a failure with a misfit value of 966.85; and the final result is unclear, with a misfit of 657.

Considering that the goal of this experiment is to determine whether increasing the number of iterations would improve the success rate of the RIS method, evaluating the unclear model is immaterial. The tests performed with 5000 iterations obtained successful results in 5 out of 20 inversions, a 25% success rate. The new experiment using 10,000 iterations achieved successful results in 5 out of 10 inversions, a 50% success rate. Ignoring the undecided inversion does not change the fact that a 200% improvement has already been achieved. Given that increasing the number of iterations from 5000 to 10,000 has already shown a 200% improvement in success rate, it is probable that increasing the number of iterations further would also improve the performance of this constraint-handling method. The fact remains however that failures exist in this data set, without any recognized cause. It is unclear if increasing the number of iterations would eliminate these failures.

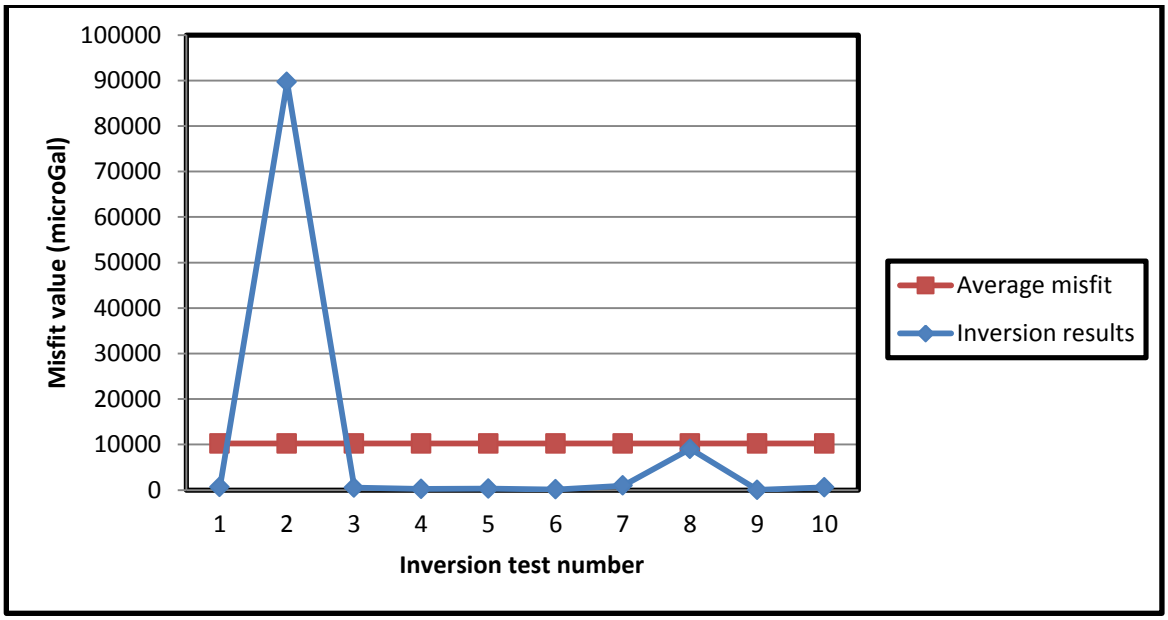

Figure 5.27: Inversion results and average misfit values of results from RIS method experiments with 10,000 iterations, outliers included.

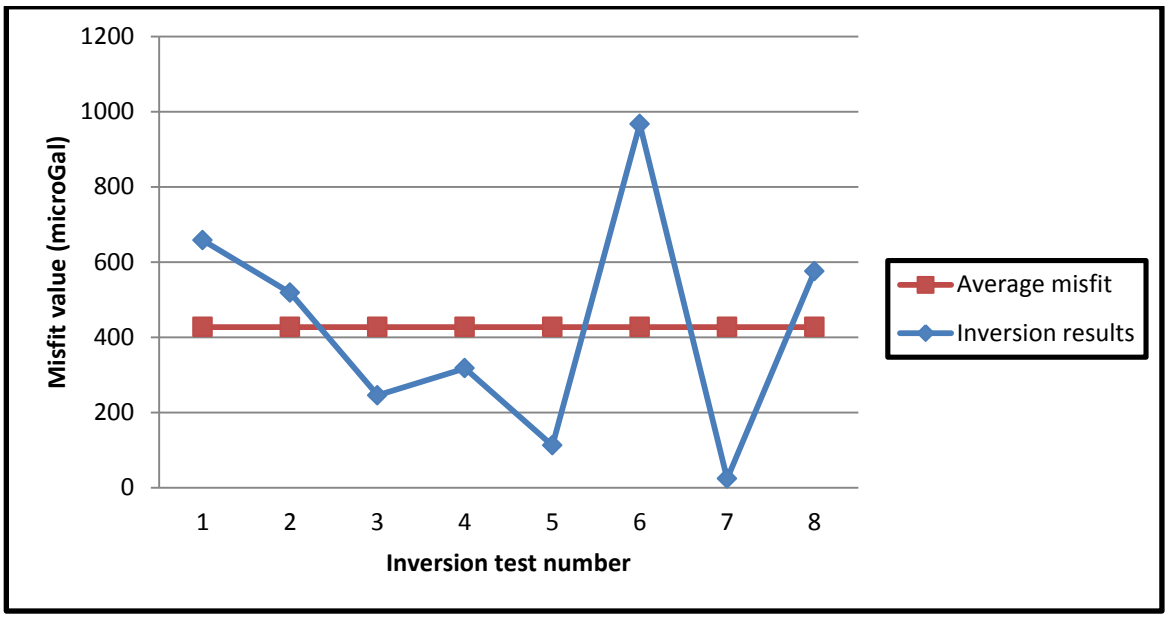

Figure 5.28: Inversion results and average misfit values of results from RIS method experiments with 10,000 iterations, outliers removed.

Table 5.3: Results obtained from inversions with sides method using the RIS method and 10,000 iterations.

10,000 iterations	Misfit	Time Taken(min)	X1	X2	X3	Z1	Z2	Z3	Rho	
1	657.69	42.49	-4.8	-3	5.14	11.42	7.43	48.71	857.42	10208.26
2	89639.49	42.5	-17.67	3.025	3.025	72.67	20.2	76.32	933.15	10208.26
3	518.18	41.9	-4.88	-0.26	5.31	36.421	5.36	36.426	2026.65	10208.26
4	245.7	41.85	-7.1	-1.64	8.6	25.07	6.89	34.42	1079.06	10208.26
5	317.5	43.72	-7.578	-2.4	9.41	24.18	6.64	34.68	952.4	10208.26
6	112.36	43.62	-3.94	-0.495	4.16	19.37	9.11	35.42	1820.51	10208.26
7	966.85	41.96	-14.11	13.93	14.01	9.85	9.847	29.23	522.2	10208.26
8	9025.45	41.75	-30	5.39	20.32	30.9	9.67	31	601.66	10208.26
9	24.11	39.55	-10.31	1.87	9.62	17.84	9.74	31.17	866.33	10208.26
10	575.25	39.65	-24.62	4.25	18.05	23.41	9.79	23.49	967.4	10208.26
average	10208.26									

Avg no outliers	427.205									

Overall the RIS method for constraint handling was unable to compete with the penalty function method described in the previous section. When used with 5000 iterations the RIS method achieved a 25% success rate, with an average misfit of 361.11 for those inversions. This method was able to prevent all models from becoming infeasible; however the overall lack of success outweighs this fact. Increasing the number of iterations doubled the success rate for this method; however the increase in performance was not enough to compare with results obtained from inversions using the penalty function.

For these reasons it is clear that for this implementation of PSO, the penalty function method is the superior method of constraint handling, and therefore it was adopted for all further examples.

5.4.4 Inversion Results of six-sided model with points method of parameterization.

As described in previous sections, implementing the points method of model parameterization into this inversion algorithm required vast changes to its coding. These changes included: introducing new methods for implementing the swarm, and new methods for detecting infeasible solutions, as well as the newly tested penalty function. The goal of the points method was to overcome the deficiencies of the sides method. This is precisely the reason for introducing a more complicated six-sided model as a test here.

Chronologically the points method, with all of its upgrades, was created prior to the testing done to determine a suitable penalty value. Due to this oversight the original points method used a smaller penalty value of 5000, which lead to failures similar to

those described in Section 5.3.1. The original approach to overcoming this obstacle was to split the particles into two groups. The first group, which contained half of the particles, was unchanged in any of its operations. The second group had a limitation placed on the maximum velocity. This set of particles could not converge towards the *gbest* value as quickly as the first group, which in theory would force the particles to act as two distinct searching units, albeit with shared knowledge. This technique was successful in overcoming the convergence issues experienced from an insufficient penalty value, as can be seen in Appendix A. Once another solution was found through increasing the penalty value to 50,000, it was decided that the new approach would be used given that it was much easier to implement and that it produced excellent results.

The average overall misfit value achieved for this suite of twenty runs, on the six-sided target shown in Figure 5.6, and constrained by a penalty function value of 50,000 was 11.39. A high value of 88.52 and a low value of 0.14 were achieved. Figure 5.29 shows a plot of these inversion results as well as the average misfit value. This plot shows the exceptional results of these tests, the highest value returned was 88.52, followed by 46.71, and then the values drop to less than a misfit of 15 microGal.

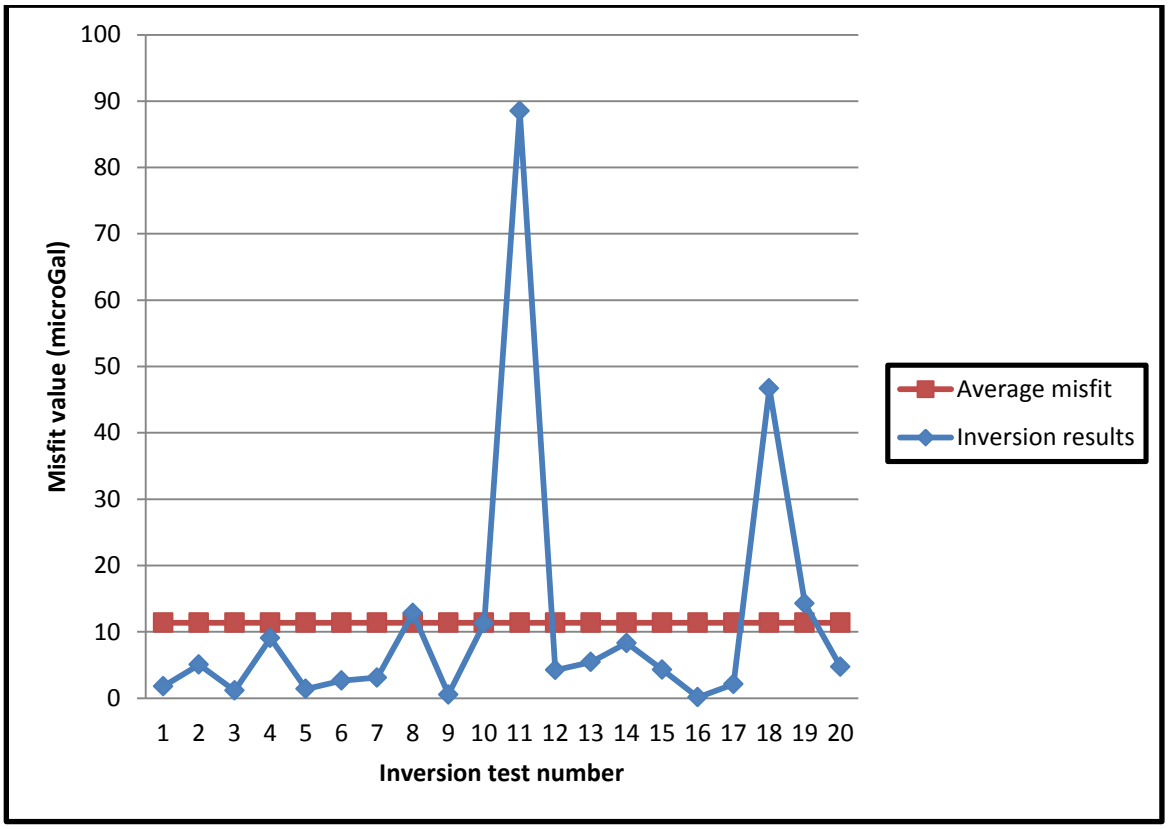

Figure 5.29: Line plot of inversion results and average misfit values from six-sided points method experiments.

Since the code was substantially modified from the previous implementations it is prudent to verify whether these models were successful, even though by previous observations they would clearly be considered excellent results.

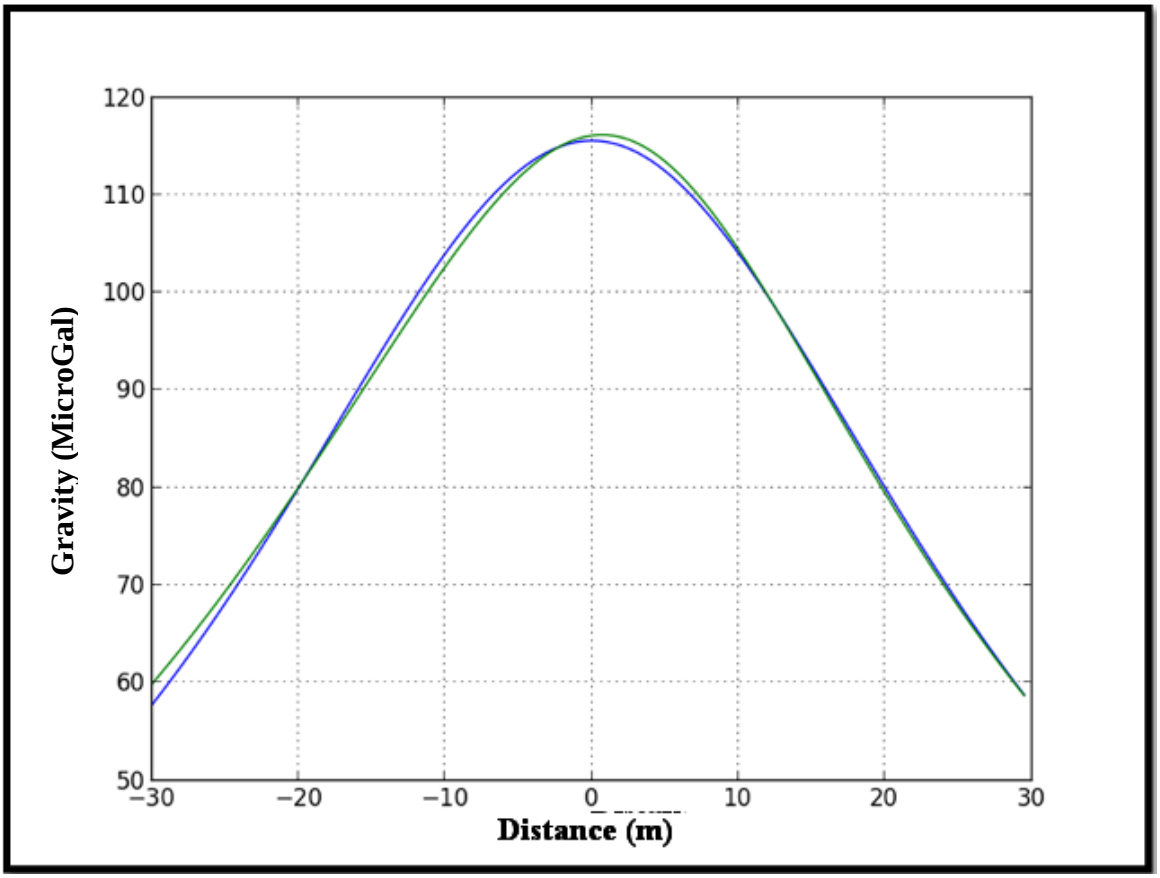

Figure 5.30: Plot of predicted data (green line) for six-sided inversion model 11 (which has a misfit 88.52); observed data are shown by the blue line.

Figure 5.30 shows that the shape and amplitude of the predicted data (green) is extremely close to the synthetic data (blue). There is some misalignment along the left of the curves, as well as the middle. The peak value of the predicted data is slightly higher and to the right of the synthetic data. Along the right however, the two curves are almost perfectly aligned. The cause of the misalignment is due to the models shape, which is seen in Figure 5.31, a plot of models 11, 16 and 18. Model 11 is closer to the surface on the right hand side than the left, causing the peak value to shift to the right. The quality of this

result is corroborated further in Figure 5.32, a data difference plot for these models. The blue curve in Figure 5.32 represents the difference in data for model 11. The difference fluctuates along the observation profile, and the fluctuations decrease from left to right. This supports the conclusions drawn by observations of the predicted data. Figure 5.32 also shows that the difference in data for model 11 is only greater than 2 microGal between x = -30m and -28m. The remaining differences are generally less than 1 microGal, which is an extremely satisfactory result.

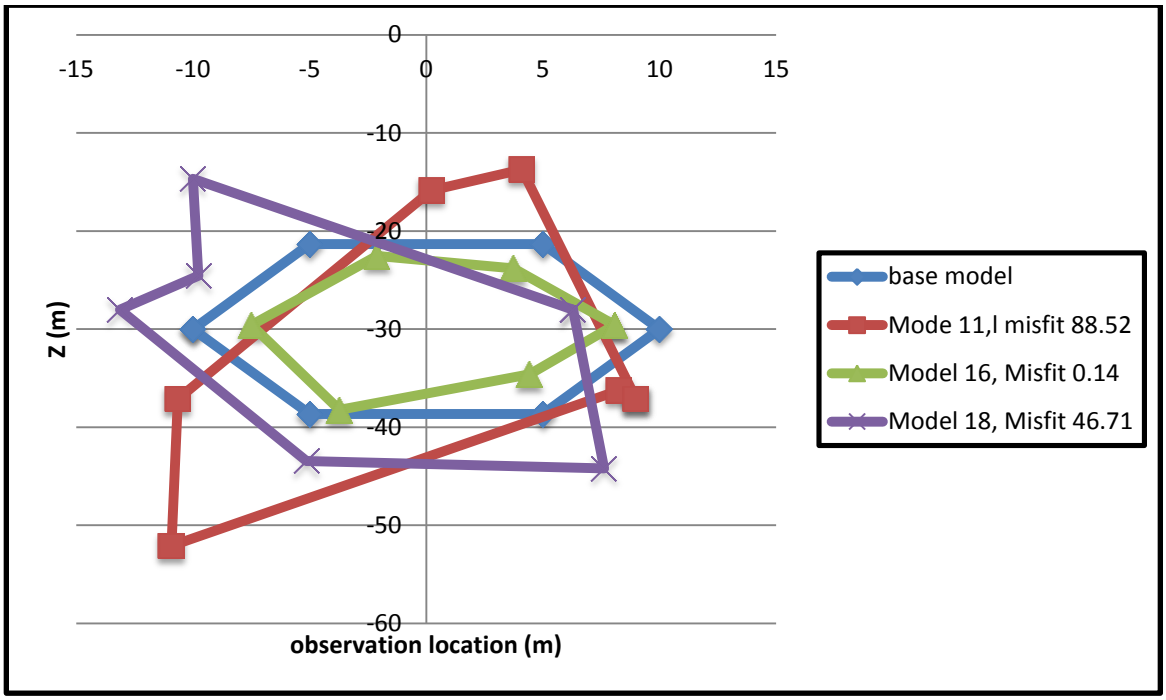

Figure 5.31: Plot of model shapes returned from tests with points method and penalty function for the six-sided examples.

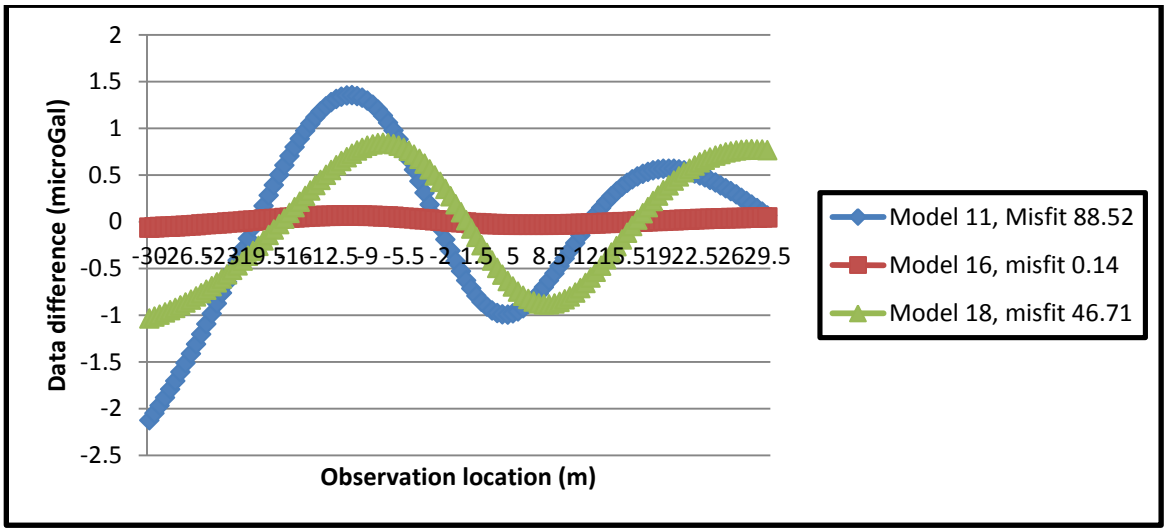

Figure 5.32: Data difference plots for models 11, 16, and 18 from tests of points method on six-sided targets.

Given that model 11 has the highest misfit value returned, and is an excellent result, it is logical to assume that the remaining results are of higher quality. This fact is reiterated in Figure 5.31, which shows model results for tests 16 and 18. Model 16 has misfit value 0.14, the lowest value returned, and model 18 had a misfit value of 46.71, the second highest in the suite. The shape of model 16 is very similar to that of the target model; it even lies inside the target. The smaller size of the model is compensated by its density value, $\rho = 1729$ kg/m^3. Model 18 on the other hand is more akin to model 11 in shape, but not positioning. The excellent quality of model 16 is seen in Figure 5.32. The difference between the predicted and synthetic data for this model is nearly zero along the length of the survey. The difference result for model 18 is also shown in this figure. Its shape is very similar to that of model 11, however its amplitudes are less, reiterating the fact that is has a lower misfit.

5.4.5 Advantages Parallel Python.

As alluded to in the preamble of this chapter, the new version of this algorithm that is being tested in this section uses a different means of optimizing the computation time used. Originally no method was used to shorten or optimize the time taken during inversions, and running a 5000 iteration inversion would take roughly 40 minutes. Later, Cython files were used to speed up the inversion. Using this method decreased the time taken to around 20 minutes for the same inversions. Parallel Python is the new method for optimizing the time used during an inversion. Table 5.4 shows that the average time taken for this suite was 19.88 minutes, however this was for 500 iterations not 5000. The new version took roughly the same time as an inversion using Cython, or half of the time it

takes to run an original inversion, or a 10,000 iteration Cython inversion. The significance of this becomes apparent when considering the parameters and number of iterations used throughout these tests. During all previous inversions the number of particles was 30, and the number of iterations was 5000. Consider each particle to be one “guess” at a model per iteration. During an original inversion there would be a total of 150,000 guesses in a 40 minute period. During a Cython inversion the time taken is cut in half, therefore the number of guesses would be double, i.e., 300,000 in a 40 minute period. The new technique uses parallel programming, which allows multiple instances of parts of the algorithm, namely the forward modeling section, to be calculated at the same time. The number of simultaneous calculations is dependent on the number of cores in the computer. The computer used had 4 cores; therefore the number of effective particles can be increased to 120, 30 for every core. Another change made to the parameters of these inversions was the number of iterations, which was decreased from 5000 to 500, due to the more complex nature of the new algorithm. The evidence of this is the average time taken, shown in Table 5.4, which was 19.88 minutes, suggesting that the new algorithm is actually slower, giving only 120,000 guesses in 40 minutes. The reason that this method is used instead of Cython is because of its potential. With a greater number of cores, a greater number of particles can be used simultaneously. For example, if 500 cores are used, the number of particles, and hence the number of guesses per iteration, can be increased to 15,000. In 40 minutes, this algorithm could perform roughly 1000 iterations, giving a total number of guesses of 15,000,000. This is a staggering number compared to

the 300,000 potential guesses made during a 40 minute period using Cython, and is precisely the reason for using this method with the new inversion algorithm.

With greater experience, comes more refined techniques. Experimenting with these inversion algorithms leads to many new ways to observe, and therefore evaluate, the inversions as they progress. One such method is a simple readout of the *gbest* misfit data at given iterations. This technique was added to the algorithm with the new additions previously mentioned, and was the driving force behind decreasing the number of iterations. The reason for this becomes clear when viewing an example readout:

Current time: 13:12:00.851000

Iteration number: 0

current gbest [[5.18765022e+00   4.50000000e+01], [7.09256413e+00 5.30952841e+01],[-6.48077994e+00   5.30569674e+01], [-6.16611076e+00 4.50000000e+01], [-7.20704283e+00   3.45603098e+01], [1.79474702e+00 4.18977651e+01], [2.15260000e+03   2.15260000e+03]]

current gbestv 11122.2509287

Current time: 13:18:05.151000

Iteration number: 100

current gbest [[4.62495810e+00   3.15383028e+01], [7.12548732e+00 3.59895095e+01],[-5.91372102e+00   3.88199711e+01],[-2.22613461e+00 3.15383028e+01],[-4.70417803e+00   2.04255299e+01],[2.01727285e+00 2.41026265e+01],[2.27406744e+03   2.27406744e+03]]

current gbestv 6.63909721567

Current time: 13:25:00.953000

Iteration number: 200

current gbest [[4.62503910e+00 3.15381913e+01],[7.12556611e+00 3.59894777e+01],[-5.91380451e+00 3.88197916e+01],[-2.22606503e+00 3.15381913e+01],[-4.70413089e+00 2.04253578e+01],[2.01736574e+00 2.41023579e+01],[2.27407196e+03 2.27407196e+03]]

current gbestv 6.63446066817

Current time: 13:31:53.423000

Iteration number: 300

current gbest [[4.62503911e+00 3.15381913e+01],[7.12556611e+00 3.59894777e+01],[-5.91380451e+00 3.88197917e+01],[-2.22606500e+00 3.15381913e+01],[-4.70413088e+00 2.04253578e+01],[2.01736576e+00 2.41023579e+01],[2.27407197e+03 2.27407197e+03]]

current gbestv 6.63445997581

Current time: 13:40:04.879000

Iteration number: 400

current gbest [[4.62503911e+00 3.15381913e+01],[7.12556611e+00 3.59894777e+01],[-5.91380451e+00 3.88197917e+01],[-2.22606500e+00 3.15381913e+01],[-4.70413088e+00 2.04253578e+01],[2.01736576e+00 2.41023579e+01],[2.27407197e+03 2.27407197e+03]]

current gbestv 6.63445997581

Current time: 13:45:28.807000

Iteration number: 500

current gbest [[4.62503911e+00 3.15381913e+01],[7.12556611e+00 3.59894777e+01],[-5.91380451e+00 3.88197917e+01],[-2.22606500e+00 3.15381913e+01],[-4.70413088e+00 2.04253578e+01],[2.01736576e+00 2.41023579e+01],[2.27407197e+03 2.27407197e+03]]

current gbestv 6.63445997581

The initial *gbest* model has a misfit of 11122.25 at iteration 0. After 100 iterations have passed, the new *gbest* value is 6.64, a vast improvement. After 200 iterations the Gbest value has barley decreased at all, to 6.63. Once 300 iterations are finished it is clear that the Gbest value has reached its lowest possible point, since the misfit is still roughly 6.63. After 500 iterations the misfit is exactly the same as for 300 iterations. This example shows that a maximum of 5000 iterations is an excessive amount, and no more than 500 iterations are needed to achieve the desired outcome.

Table 5.4: Results obtained for inversions using the points method, with penalty value 50,000 on a six-sided model.

Test #	Misfit	Time Taken(min)	X1,Z1	X2,Z2	X3,Z3	X4,Z4	X5,Z5	X6,Z6	Rho
1	1.799	19.07	6.96,31.019	2.529,38.431	-5.339,32.649	-5.826,31.018	-3.94,22.15	3.94,25.88	2105.422
2	5.069	19.23	3.702,30.08	6.66,39.74	-4.52,38.34	-7.11,30.08	-1.44,22.95	3.914,18.51	1592.49
3	1.158	19.78	4.33,35.35	4.83,37.31	-7.11,38.00	-3.83,35.35	-8.20,25.64	6.923,22.25	1656.79
4	9.08	19.785	3.97,35.45	-0.733,31.27	2.72,45.02	-8.86,35.46	-0.977,20.09	7.55,23.72	1376.77
5	1.41	19.54	9.74,29.23	5.09,32.44	-4.04,34.78	-9.93,29.22	-1.12,24.04	5.64,25.24	2041.78
6	2.65	19.53	6.21,31.65	4.82,37.19	-2.06,37.91	-4.63,31.65	-5.33,22.04	4.15,25.19	1978.93
7	3.105	20.695	6.58,26.25	6.80,35.27	-3.75,35.195	-7.098,26.25	-2.46,22.07	-0.61,25.34	2047.7
8	12.83	20.69	2.59,25.21	5.86,42.53	-7.63,31.95	-6.66,25.21	-0.685,25.36	9.13,14.29	1779

9	0.55	20.05	10.25,31.34	7.10,36.09	-6.31,40.196	-8.66,31.34	-4.78,21.74	2.20,19.91	1056.195
10	11.32	20.05	9.80,31.12	5.35,37.31	-1.00,37.64	-6.69,31.16	-2.41,18.14	1.85,27.14	1715.92
11	88.52	20.195	8.99,37.11	8.27,36.25	-10.93,52.095	-10.65,37.11	0.225,15.87	4.09,13.47	692.88
12	4.25	20.195	9.62,29.62	3.78,34.39	-2.297,35.41	-5.64,29.62	-4.49,22.08	3.18,27.80	2543.51
13	5.45	20.19	5.05,30.75	4.54,40.77	-5.52,39.97	-5.97,30.75	-1.03,19.61	3.68,20.99	1433.64
14	8.33	20.195	13.22,28.72	4.09,38.33	-5.37,35.08	-10.65,28.72	-2.28,18.96	3.64,25.36	1109.54
15	4.33	19.765	4.34,29.86	1.83,36.94	-4.19,39.64	-5.17,29.98	-0.52,23.095	3.93,21.86	2288.6
16	0.14	19.77	8.07,29.58	4.43,34.62	-3.73,38.22	-7.495,29.58	-2.12,22.54	3.74,23.80	1729.95
17	2.12	19.85	8.01,26.39	3.06,36.82	-3.67,39.94	-8.49,26.39	-1.88,23.49	5.44,22.18	1442.72
18	46.71	19.85	6.29,28.04	7.62,44.19	-5.13,43.44	-13.13,28.03	-9.77,24.57	10.00,14.69	659.02
19	14.29	19.62	3.52,38.62	7.499,44.85	-1.67,42.37	-4.07,38.62	-7.82,29.59	4.38,16.24	1315.35
20	4.76	19.62	14.49,19.31	-2.71,45.13	-11.27,24.21	2.75,19.31	1.09,21.52	6.21,23.45	896.295

average	11.39355	19.8835							1573.125

Table 5.4 continued

5.5 Conclusions.

This chapter began by introducing the points method for model parameterization, and in doing so it indicated areas of the code which needed to be changed, and areas which did not. Section 5.1 described the changes that were made to the algorithm to make the points method possible. These changes include the swarms shape and size, methods for initializing the swarm, and new techniques for detecting infeasible solutions. This section also introduced the six-sided model, which was used to test the points method. Section 5.2 explained the two different constraint handling methods which were experimented with in this chapter, namely, the penalty function method and the RIS method. Section 5.3 illustrated the results obtained from the experiments with these constraint handling methods. The first experiment, in Section 5.3.1, used the sides method, with a penalty function value of 500. Using the same model and parameterization method allows the data to be compared directly, and thus to determine if the penalty function or the RIS method would be superior. These tests showed that a penalty value of 500 was unable to prevent infeasible models, and was ultimately deemed unsuccessful. Section 5.3.2 described the results obtained with an increased penalty value of 50,000. This increase in penalty value greatly improved the results returned. The results contained one poor model, however it was discovered that an error in coding was responsible for the large misfit not ineffectiveness in the penalty method. Section 5.3.3 continued by describing the tests done using the RIS method. These results were less than optimal, with three distinct subsets of quality. The first set was considered successful inversions, and contained 5 tests results. The second and third sets were all unsuccessful inversions. The

cause of the failures was not determined however further tests with an increased number of iterations were performed. Upon increasing the number of iterations, the number of successful results increased by 200%; however the cause of the failures was still not isolated or eliminated. Overall this method was not able to compete with results obtained from the penalty function; therefore no further tests were carried out with the RIS method. Section 5.3.4 described the results of tests performed with the points method as well as the upgrades to the algorithm. These were the best results obtained thus far. The results all had misfit values less than 88. This section then considered the difference in using Parallel Python and Cython. The explanation showed that using Parallel Python has a much greater potential than using Cython, which is the reason for its adaptation in the final version of the inversion algorithm. This section concluded by describing the reasoning behind the decrease in the number of iterations used, from 500 to 5000. An example read out of an inversion showed that using 5000 iterations was excessive, and no information was gathered after roughly 300-400 iterations.

Chapter 6 will experiment further with the points method, testing objects with larger number of sides, as well as using noisy data. This will demonstrate the robustness of this algorithm, and emphasize the fact that PSO can be used for non-linear inversions of geophysical data.

Chapter 6 : Further Testing of Points Method

6.1 Introduction

This chapter will investigate objects with ten sides, with and without noise added to the synthetic observed data. The inversion algorithm used is the culmination of all the previously experimented versions, and is the best version found thus far. The code uses the points method of model parameterization with the technique of initializing the swarm described in Section 5.1. Finding infeasible models and boundary infractions is done using the two part method also described in Section 5.1. The constraint handling method used is the penalty function, since it had the best test results overall when used with a penalty value of 50,000. This version will use Parallel Python as opposed to Cython, for the reasons given in Section 5.3.4, and will have a maximum number of iterations of 500. The stopping parameter, which was triggered for misfit values less than 0.1, was removed for this version of the algorithm. Adding noise to the data changed the lowest misfit values achievable for every data set, rendering the previous parameter obsolete. All other areas of the code, that is, the misfit function and the forward method are unchanged from the implementation used in the last chapter. The number of particles used is 120, as was established in Section 5.4.4. The testing procedure will remain unchanged: performing 20 inversions and recording the coordinates of the points, the density in kg/m^3, the final misfit value, and the time taken. (The source code can be found in Appendix B, along with all results not shown in this chapter)

6.2 Ten-sided inversion target, and noisy data creation.

Creating the ten-sided inversion target consisted of picking 10 points around a circle with a radius of 10 m and centered at X = 0.0m, Z = -40.0m. The density value of this model remains 1000 kg/m^3. Figure 6.1 is a diagram of the ten-sided inversion target, displayed inside the survey region.

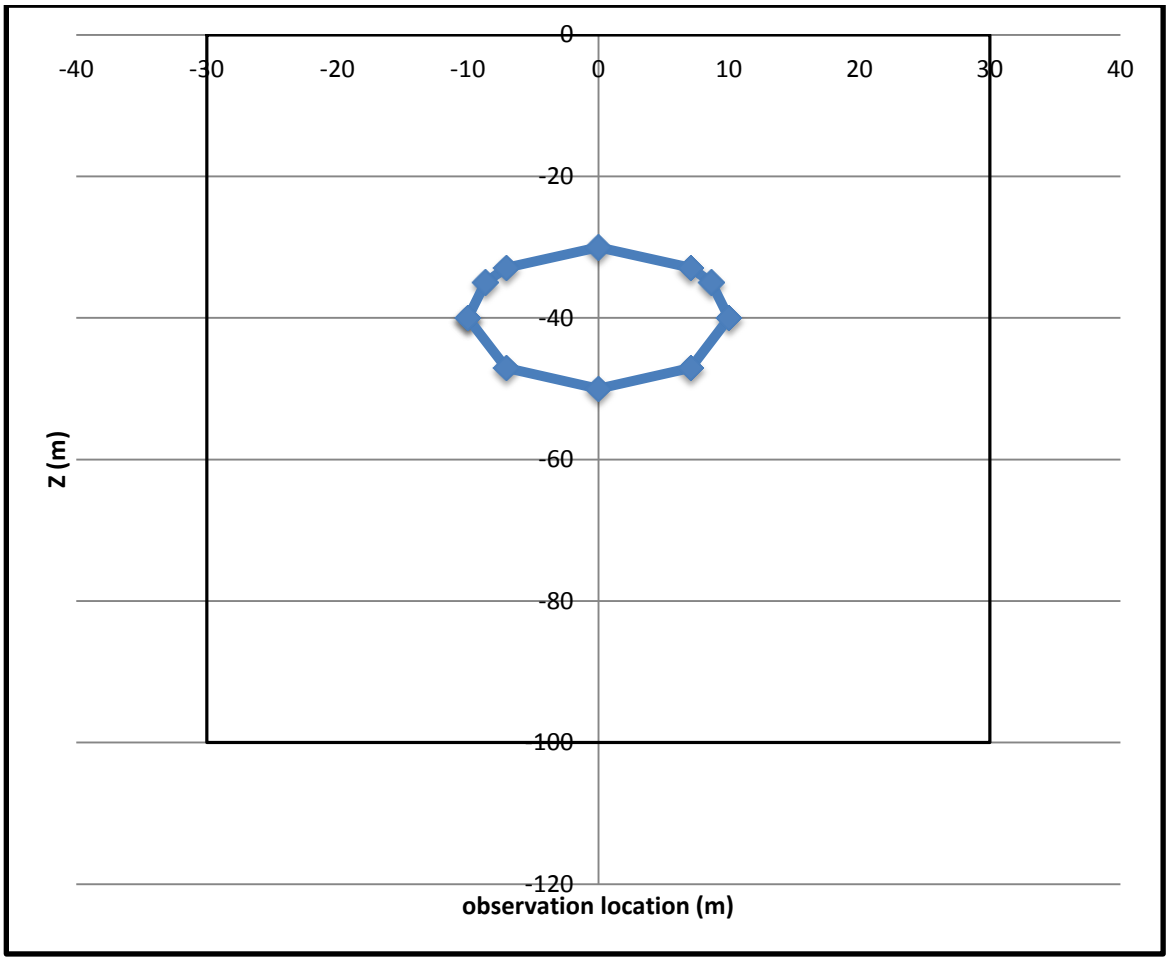

Figure 6.1: Ten-sided inversion target (blue) inside the survey region, which is outlined in black.

Figure 6.2 is a diagram of the six-sided model (red), and the ten-sided model (blue), with the synthetic data from both plotted in pink and blue respectively. The synthetic data

calculated for the ten-sided model has a broader shape than that for the six-sided model. This is because it is buried deeper.

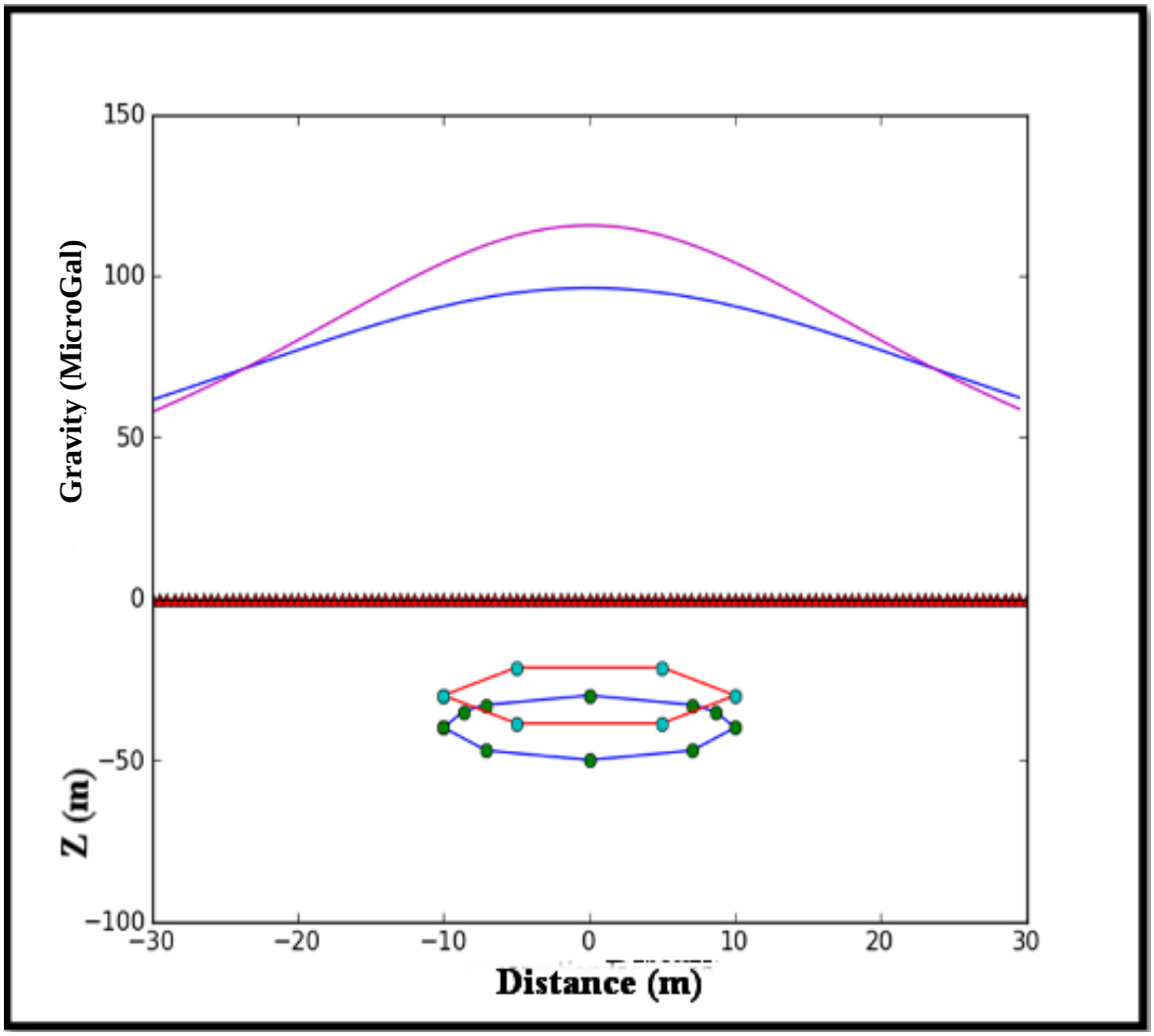

Figure 6.2: Diagram of the six- and ten-sided models, with their synthetic data plotted above. Six-sided model is in red, with pink gravity curve. Ten-sided model is blue with blue gravity curve.

Creating noisy data increases the difficulty of the inversion, thus further testing the limits of the algorithm. The method for adding noise to the data used here is straightforward. It consists of adding Gaussian noise to each data point using a built in Python Gaussian

random number generator called Random.Gauss. This application has two inputs: the mean, mu, and the standard deviation, sigma. Three sets of noisy data were created, all with a mean value of 0, but with increasing standard deviation values. The data with low noise used a standard deviation of 1 microGal, medium noise had a standard deviation of 3 microGal and high noise had 10 microGal. Figure 6.3 is a plot of the noisy data sets and the synthetic data without noise added. The low noise data seen in red is slightly perturbed, with random peaks and valleys. The medium noise data in blue has higher amplitude in its perturbations, but the general shape of the plot is still maintained. The yellow curve is the high noise data; it has a very large variation with position, and the shape of the curve is mostly masked.

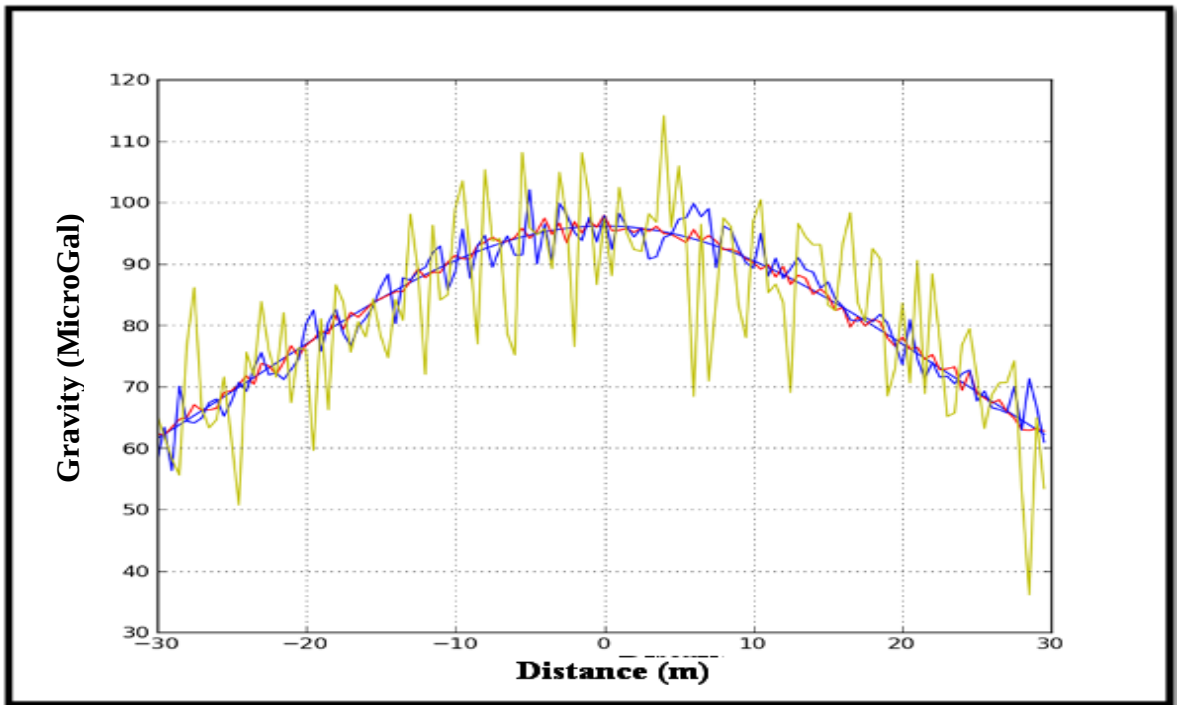

Figure 6.3: Synthetic data plots for no noise (smooth blue), low noise (red), medium noise (jagged blue) and high noise (yellow).

6.3 Results

6.3.1 Results for ten-sided inversions of noise free data.

The results of the ten-sided inversions of the noise free data were on par with those produced from inversions on the six-sided model. The average misfit achieved was 6.77 with high and low values of 80.71 and 0.0043 respectively. Figure 6.4 is a graph of the results (blue) obtained along with the average misfit value (red). It is clear from this plot that these are exceptional results, with only one test above a misfit of 20 while the majority of the values fall below the average value of 6.77.

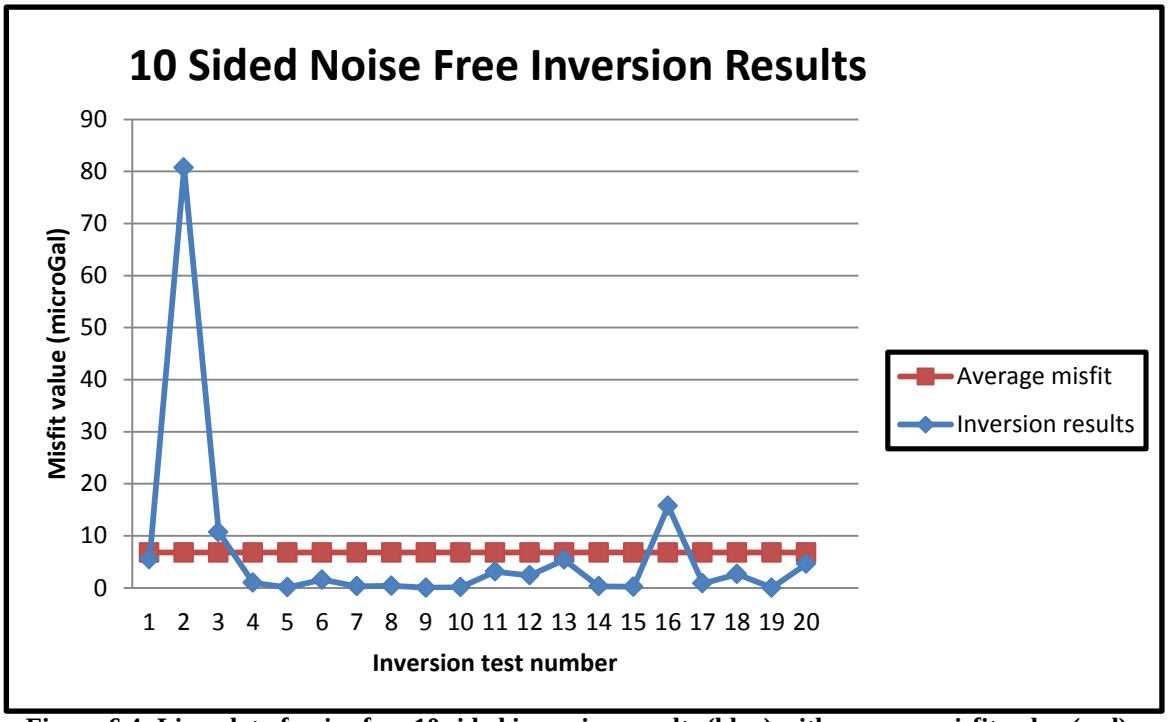

Figure 6.4: Line plot of noise-free 10 sided inversion results (blue) with average misfit value (red).

These results can be evaluated using the same criteria established in Section 5.4.4, since the same algorithm is used and that set of inversions had a higher overall misfit and average misfit. The results from that section were deemed extremely successful, and hence these results are also exemplary. This is confirmed in Figure 6.5, a plot of the predicted data from test 2, the highest misfit returned in this set. The fact that these results had lower misfits overall is also noteworthy since a higher number of sides was used, which increases the difficulty of the inversions.

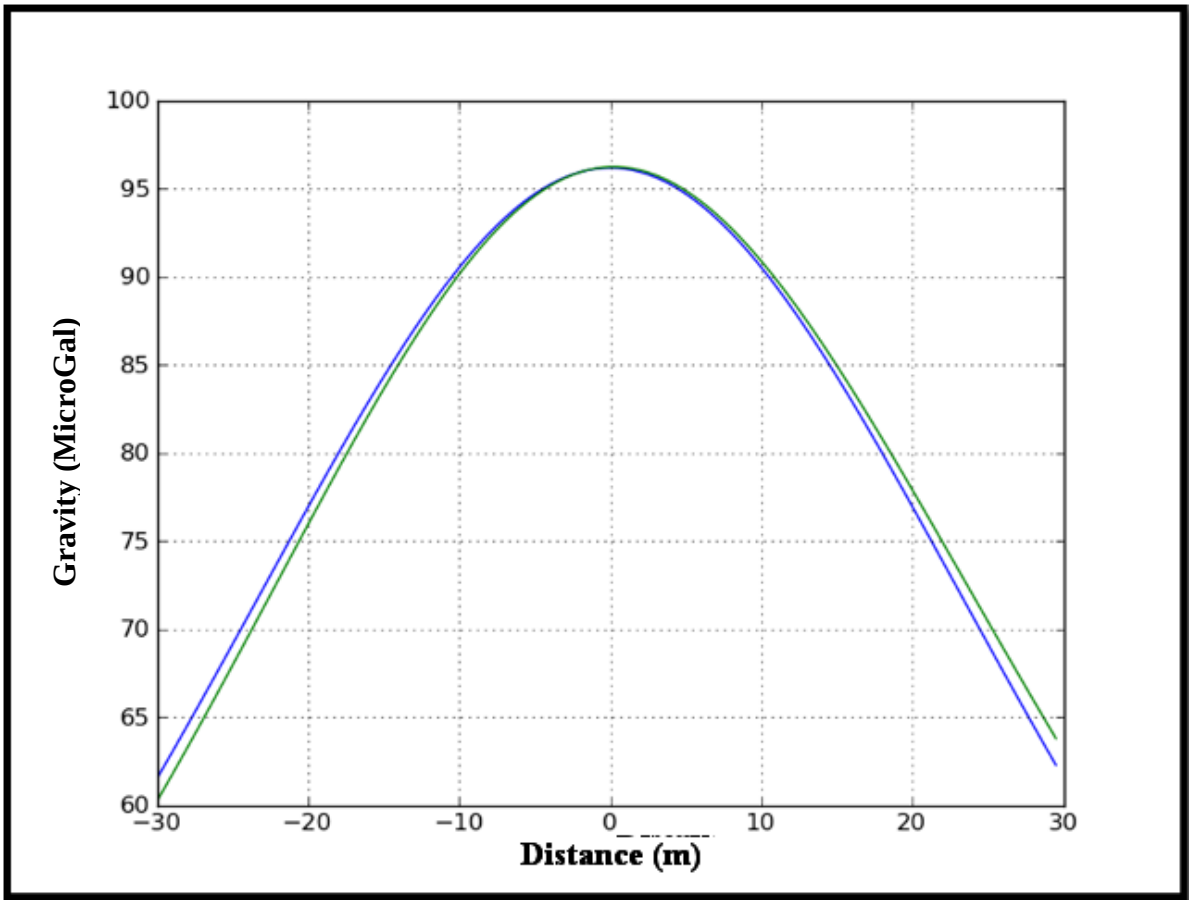

Figure 6.5: Plot of predicted data(green) for run 2 (which has a misfit of 80.71) of the noise-free ten-sided inversions. The blue curve shows the noise-free observed data.

6.3.2 Results for ten-sided inversions using medium noise.

The results in this section will be for medium noise only; the results of low and high noise are recorded in Appendix A. The inversions using medium noise returned excellent results. The average misfit for this set was 975.30, which when compared to previous results would indicate one or more infeasible solutions. This is actually caused by the jaggedness in the data and not some flaw in the inversion process, which is demonstrated in Figure 6.6, a plot of the medium noise data and the original noise free data.

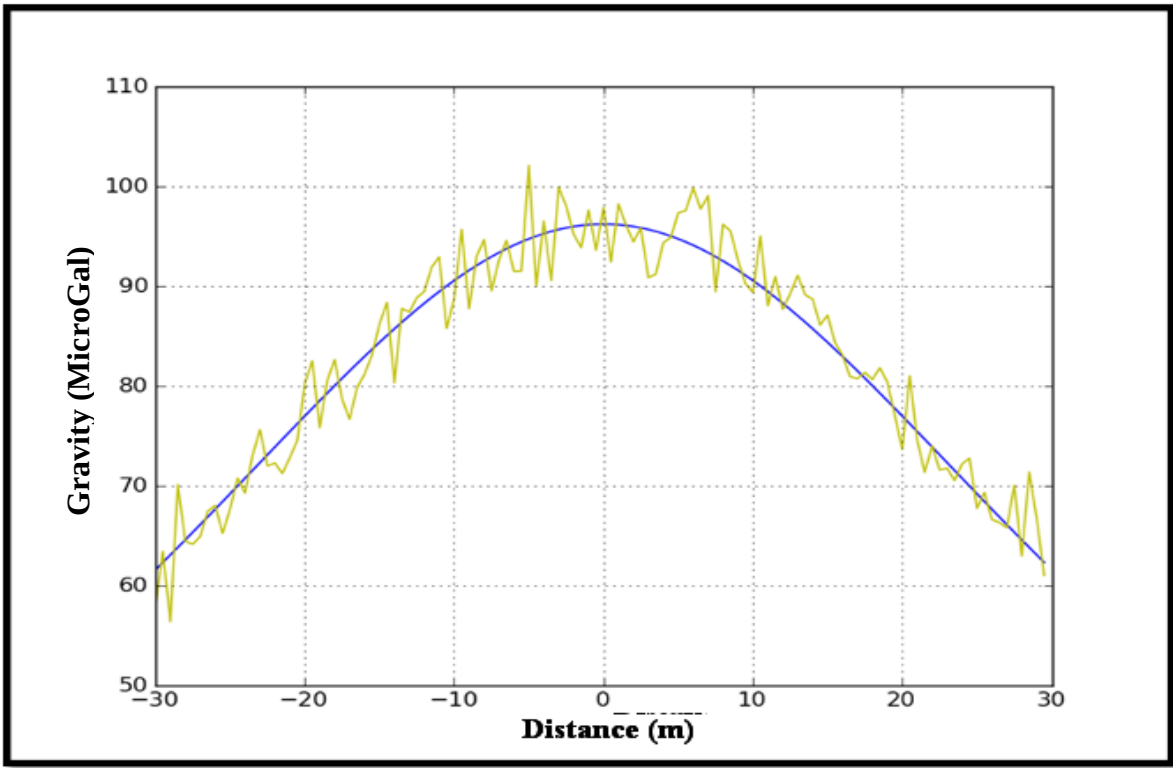

Figure 6.6: Plot of the noise-free synthetic data (blue) with the medium noise data (yellow).

Performing a misfit calculation between the medium noise data and the noise free data gives a misfit value of 998.99, which is close to the average value returned. Once this

value is considered, the results can be viewed in a new light. The highest misfit value returned in this set was 1103.16, from test 1, whereas the lowest misfit returned was 956.51, lower than the misfit value for the synthetic noisy data. This is not an error, and can be explained when viewing Figure 6.7, a plot of the predicted data for this model, along with the noise free and medium noise data sets, shown in green, blue and yellow respectively.

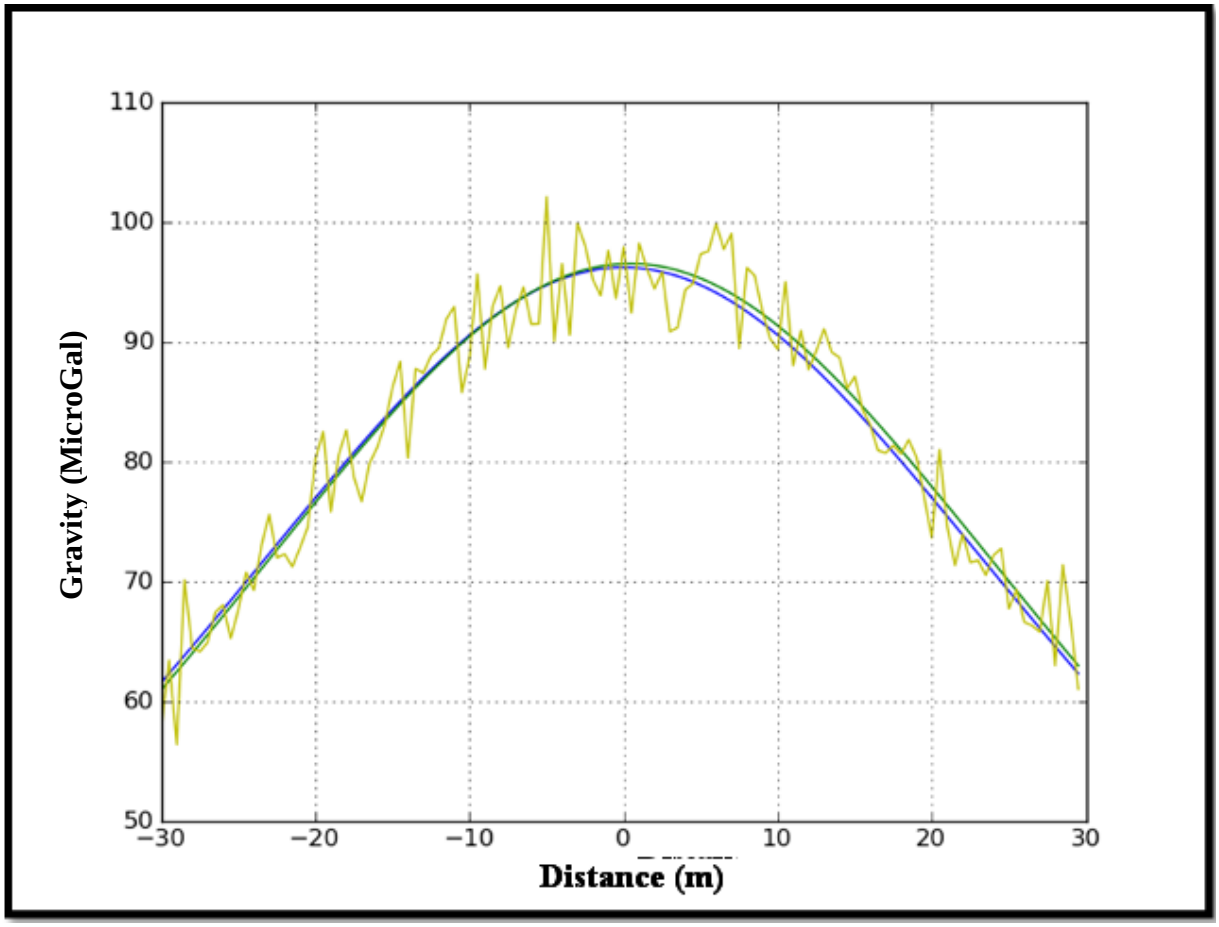

Figure 6.7: Plot of predicted data for model 11, (misfit 956.51) (green), with noise free (blue) and medium noise (yellow) data sets.

As described in Section 6.2, the noisy data sets were generated by adding random Gaussian noise to the noise free synthetic data set. Since the “noise” mostly falls between plus and minus the standard deviation, values can be randomly added or subtracted from the synthetic data to produce the new sets. As a result, the medium noise data has an area along the right side were more points fall above the synthetic data curve than below, creating a higher trend in the data, seen in Figure 6.6. The inversions are therefore attempting to fit this trend, which can be seen in Figure 6.7. The blue synthetic and green predicted curves line up well along the left side, but towards the middle and right the predicted curve has fit the new higher trend. This raises it above the synthetic data slightly, producing a lower misfit as a result.

The model with the highest misfit in this set was number 1. The predicted data from this model are seen in Figure 6.8 along with the noise free and medium noise data sets, green, blue and yellow respectively. This figure is nearly a mirror image of Figure 6.7. The predicted curve is slightly higher on the left side, near the middle the curves line up, and along the right side there is little separation. Both curves pass though the yellow data set near the middle of every peak and valley, and since one curve is the original data set, and they are so closely matched it is only logical that this would be a successful inversion. Consequently, the remaining results would also be considered successful, given that they all have lower misfit values.

Figure 6.9 is a diagram of the models returned in tests 1 and 11 and the base model. Both models have irregular shapes that do not resemble the circular base model.

This is not unexpected since the only restraint on the inversions is a misfit function which does not guide the model shapes in any way, simply minimizes the misfit of the model. Even though the models are irregular and jagged shapes, they are not flattened out in the horizontal or vertical directions. In fact they are generally positioned in the same area as the base model with centers that are very close to that of the base model.

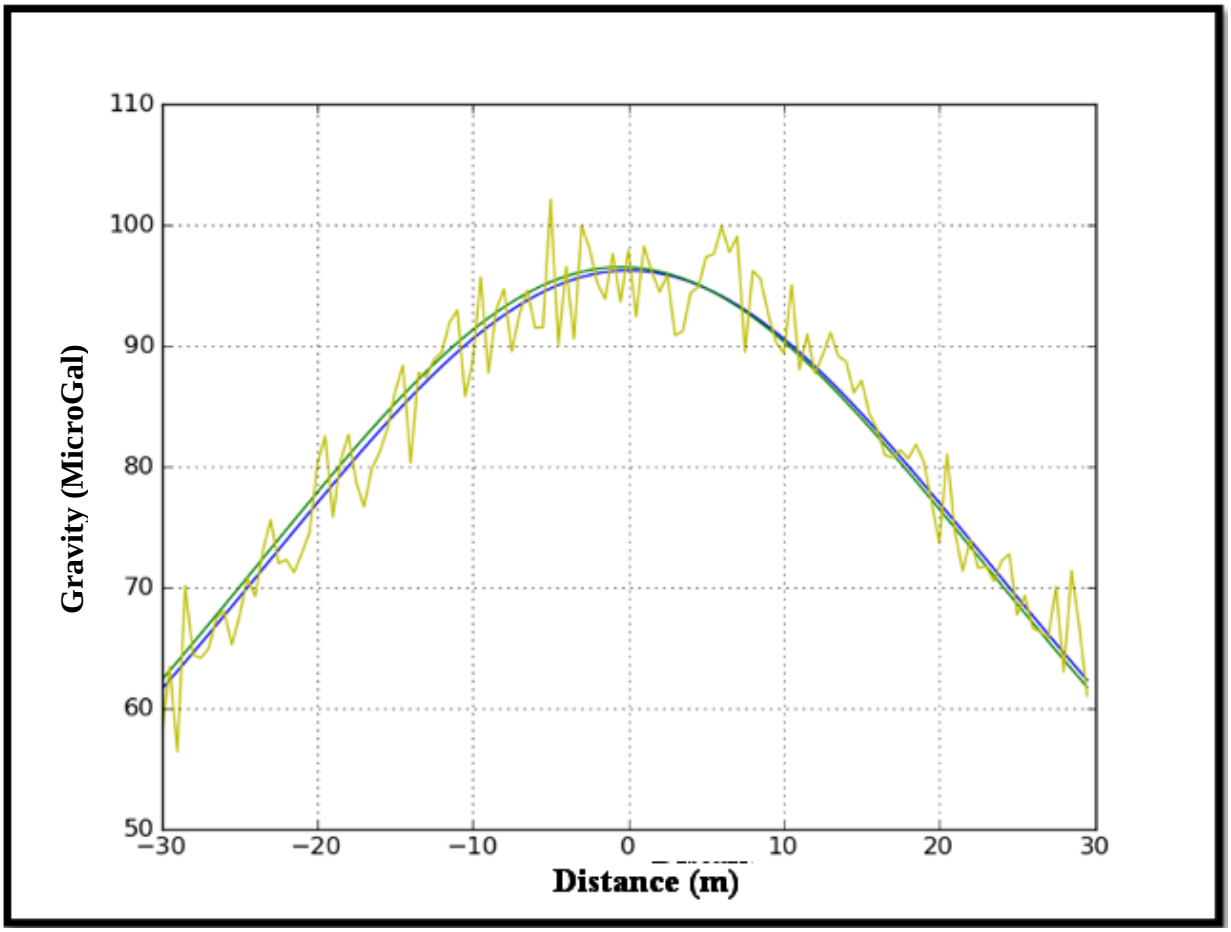

Figure 6.8: Plot of predicted data for model 1, which has a misfit 1103.16 (green), with noise free (blue) and medium noise (yellow) data sets.

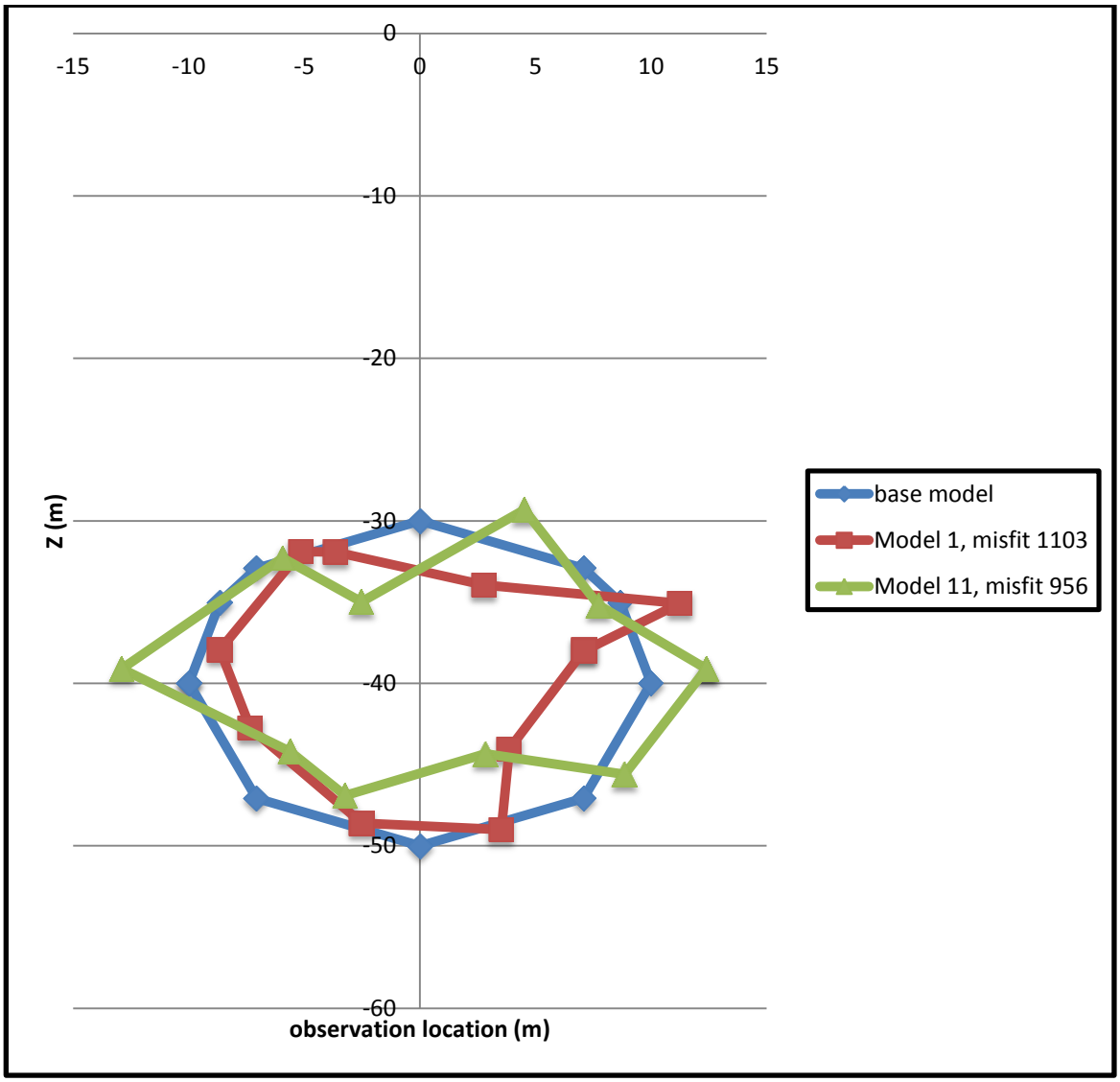

Figure 6.9: Diagram of models 1 and 11, with base model for reference.

6.3.3 Inversion time differences

One noteworthy difference between inversions of six and ten-sided models was the average time to completion. Table 5.4 shows that for six-sided models the average time to completion was 19.88 minutes, which was obtained without any tests having achieved

stop parameters. The average time to completion for inversions using data with medium noise was 32.27 minutes, nearly a 13 minute increase. This increase in time is expected since the number of sides is practically doubled, meaning each iteration would be more computationally intensive, and thus take longer. It also cannot be ignored that given the randomness of the noisy data a stop parameter was not used for the noisy data sets, and given that the average misfit returned was lower than that of the misfit between the noise free and medium noise data, this would have been triggered several times, decreasing the overall average time.

6.4 Conclusions

This chapter began with an introduction of the latest algorithm being used, which was the successful result of the experiments from the previous chapters. The ten-sided model used in this chapter was created in order to increase the difficulty of the inversion but also to verify that the algorithm operates with the increased amount of sides. The model itself was circular in shape, with a radius of 10 m, centered at X = 0m, Z = 30m. The noisy data was created by adding random Gaussian noise to the synthetic data, which was done with a built in Python package. The package uses two inputs, the mean, which was set to zero, and the standard deviation, which was set to 0, 3, and 10 for the low, medium and high noise examples respectively. The inversion results for the noise free data were excellent, with an average misfit value of 6.77, the best of all test sets. Adding the medium noise to the data set didn’t affect the quality of the results. The average misfit returned was lower than the misfit calculation of the synthetic data and medium noise, which was not an error, but was caused by a new trend created by the noisy data. Overall the algorithm

performed remarkably well, with and without noisy data, retuning outstanding results for every test, with the sole down side being increased computation time.

Chapter 7 : Conclusions

This thesis described the process of creating an inversion algorithm for performing non-linear inversion of multifaceted polygons using particle swarm optimization (PSO). This work demonstrated that using PSO for geophysical inversions is a viable option for conducting non-linear inversions for complex objects. The algorithm created here generally followed the recommended parameters of Carlisle and Dozier (2001), namely 30 particles with a global neighborhood. The particles are updated synchronously, with a cognitive-to-social ratio of 2.8 to 1.3 for φ magnitude of 4.1**.** Clercs constriction factor, K, was used since this leads to the greatest improvement in performance. There was no constraint placed on V_{max}.

Each chapter of this thesis described a new development in the process of creating an inversion algorithm. The first tests in Chapter 3 were to minimize several benchmark functions, namely the Sphere function, the Rosenbrock function, Rastrigins function, the Griewangk function and the Schaffer F6 function. The results obtained showed good convergence percentages, with 100% success in many cases, and no less that 85% convergence overall. The algorithm also showed excellent behavior in the three examples with multiple local minima. Avoiding the local minima successfully is exactly the sort of behavior sought from PSO.

Chapter 4 expanded on the difficulty of the tests by incorporating the sides method of model parameterization and forward modeling into the software. The tests were first conducted on four- then five-sided models without constraints. The five-sided unconstrained inversions failed to return models with acceptable shapes in many of the 20

test runs, leading to the creation of the corner switching method for constraint handling. The corner switching method showed great improvement in the four- and five-sided models, but it was ultimately deemed to be over-complex, which led to the investigation of new constraint handling methods in Chapter 5.

Chapter 5 described the investigation of two new constraint-handling methods: the penalty function method and the rejection of infeasible solution (RIS) method. The penalty function method was found to be superior to the RIS method when an appropriate penalty value (50,000) was used, returning better average results in fewer iterations. This penalty function method was then applied to the new version of the software, which also had a new method for model parameterization, namely the points method, among other changes. The points method proved to be the best version of the algorithm tested up to that point, returning excellent results.

Chapter 6 was designed to increase the difficulty of the inversion in order to further test the viability of the software, by increasing the number of sides of the target model, and by adding noise to the data. The inversion results for the noise free data were excellent, with an average misfit value of 6.77. Adding medium noise, with a standard deviation of 3 microGal, to the data set didn’t affect the quality of the results. The sole down side found was increased computation time with increased number of sides.

The algorithm created during the course of this thesis showed that using PSO for non-linear inversions of multisided polygons is an effective and viable approach to inverting geophysical data, and could be adapted further with great potential.

Bibliography

Battiti, R., Brunato, M., & Pasupuleti, S. (2005). Do not be afraid of local minima: Affine shaker and particle swarm.

Blakely, R. J. (1996). *Potential theory in gravity and magnetic applications*. Cambridge University Press.

Carlisle, A.; Dozier, G. (2001). "An Off-The-Shelf PSO". *Proceedings of the Particle Swarm Optimization Workshop*. pp. 1–6.

Coath, G., & Halgamuge, S. K. (2003, December). A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (Vol. 4, pp. 2419-2425). IEEE.

Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on* (pp. 39-43). IEEE.

Hu, X., Eberhart, R. C., & Shi, Y. (2003, April). Engineering optimization with particle swarm. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE* (pp. 53-57). IEEE.

John D "Parametric Equation of a Circle", Math Open Reference. Retrieved 7 May 2013 from http://www.mathopenref.com/coordparamcircle.html

Kennedy, J., & Eberhart, R. (1995). November.“Particle swarm optimization”. *Neural Networks*, 1942-1948.

Laskari, E. C., Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization for integer programming. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (Vol. 2, pp. 1582-1587). IEEE.

Li, Y. & Oldenburg, D.W., 1996. 3D inversion of magnetic data, Geophysics,61,394–408

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*.

Pierce, Rod. (21 Apr 2013). "Polygons". Math Is Fun. Retrieved 7 May 2013 from http://www.mathsisfun.com/geometry/polygons.html

Shaw, R., & Srivastava, S. (2007). Particle swarm optimization: A new tool to invert geophysical data. *Geophysics*, *72*(2), F75-F83.

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 3). IEEE.

Yuan, S., Wang, S., & Tian, N. (2009). Swarm intelligence optimization and its application in geophysical data inversion. *Applied Geophysics*, *6*(2), 166-174.

Appendix A

Test #	Misfit	Time Taken(min)	X0,Z0	X1,Z1	X2,Z2	X3,Z3	X4,Z4	X5,Z5	RHO
1	1009.44	48.5	[5.95,33.65]	[4.50,43.27]	[-2.88,31.66]	[-10.09,33.65]	[-3.67,24.33]	[2.66,31.01]	1780.34
2	1005.86	49.23	[5.29,26.72]	[1.18,39.56]	[-5.35,34.38]	[-2.23,26.72]	[-4.41,22.16]	[5.98,26.56]	2619.58
3	989.49	52.03	[1.34,28.85]	[6.23,45.40]	[-5.60,35.14]	[-4.75,28.85]	[2.21,17.62]	[3.74,27.22]	1931.91
4	980.23	47.56	[10.33,36.10]	[-0.68,41.10]	[7.35,50.94]	[-11.14,36.10]	[2.38,13.86]	[4.59,25.58]	894.1
5	989.06	49.33	[4.82,32.22]	[0.73,39.55]	[-9.11,41.71]	[-2.71,32.22]	[-2.34,19.58]	[4.89,24.44]	1867.22
6	980.72	50.03	[3.82,32.02]	[-0.32,40.05]	[-0.02,46.92]	[-8.24,32.02]	[-4.29,30.78]	[4.07,16.07]	1826.57
7	1000.08	48.45	[2.09,31.56]	[6.67,36.69]	[-11.07,32.89]	[-5.27,31.55]	[-9.84,23.48]	[6.75,25.47]	2634.74
8	996.03	47.41	[6.14,34.29]	[6.07,34.70]	[-1.36,36.08]	[-10.39,34.29]	[0.66,22.49]	[4.76,22.25]	1890.73
9	985.81	47.13	[-3.82,38.85]	[3.36,37.76]	[-1.93,41.92]	[-6.24,38.85]	[2.15,21.33]	[5.84,25.36]	2826.48
10	991.14	49.35	[[2.88,31.84]	[1.48,38.42]	[-2.80,43.33]	[-3.50,31.84]	[-1.42,22.55]	[3.23,21.82]	2666.09
11	1004.59	49.06	[7.05,29.23]	[0.58,30.08]	[-1.66,40.85]	[-8.64,29.23]	[-8.55,28.37]	[8.28,22.27]	2190.66
12	994.06	48.39	[8.48,29.46]	[2.59,34.86]	[-9.98,40.19]	[-3.03,29.45]	[-7.11,19.66]	[6.31,29.24]	2089.27
13	1007.65	49.44	[10.25,28.54]	[2.44,30.42]	[0.99,39.66]	[-8.79,28.54]	[-2.73,26.68]	[-0.34,22.97]	2268.78
14	981.65	47.51	[4.15,29.31]	[1.19,42.85]	[-4.72,39.37]	[-7.20,29.31]	[-3.49,28.43]	[4.16,16.82]	1723.56
15	990.23	48	[6.13,31.92]	[2.83,36.35]	[-7.04,36.17]	[-11.58,31.92]	[-1.24,27.45]	[4.00,17.56]	1780.95

16	995.29	49.03	[7.36,30.70]	[4.97,34.68]	[-5.49,39.20]	[-8.61,30.70]	[-1.36,23.35]	[4.31,20.94]	1515.1
17	984.95	48.4	[2.00,32.74]	[3.22,44.72]	[-3.72,47.54]	[-0.53,32.74]	[-6.21,27.77]	[3.78,17.74]	2029.17
18	977.89	48.38	[-2.54,28.79]	[5.33,35.06]	[-1.91,48.79]	[-3.53,28.79]	[-7.13,26.52]	[8.33,15.52]	2084.03
19	982.17	48.04	[5.84,30.27]	[6.04,35.30]	[-5.76,45.80]	[-6.38,30.27]	[-4.64,26.16]	[2.93,17.01]	1251.16
20	994.35	47.01	[14.33,34.28]	[-0.08,40.30]	[-4.25,40.99]	[-8.17,34.28]	[0.86,16.35]	[7.37,35.74]	1258.61