INVERSE PROBLEMS USING GRAVITY GRADIOMETRY

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2018

David Leahy
School of Mathematics

Contents

Word count 57660

List of Tables

List of Figures

The University of Manchester

David Leahy
Doctor of Philosophy
Inverse Problems Using Gravity Gradiometry
October 30, 2018

In this study we explore the feasibility of reconstructing the density distribution of a cargo container to an appropriate degree using gravity gradiometry with the aim of identifying the presence of fissile material.

The resulting inverse problem is highly ill-posed with a large null space and so several techniques are used to attempt to achieve accurate and reliable solutions. A level set method is employed to model the sharp discontinuities expected, built on the use of radial basis functions.

Local and global optimisation methods are explored, including sparsity, gradient descent and a genetic algorithm. Ultimately two optimisation methods, the method of steepest descent and a genetic algorithm, are combined in an ensemble hybrid algorithm that utilises information obtained about the eccentricities of this particular problem to produce many solutions. The solutions are analysed in an effort to provide an indicator for the presence of high density material whilst avoiding both false positive and negative results.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

Acknowledgements

I would like to thank my supervisor Oliver Dorn for his support. His knowledge on the topics covered in this thesis has been instrumental over the last 4 years.

I would also like to thank my industrial sponsor for their support and for providing me with an interesting problem such as this.

I would like to thank my family for being supportive of me during all of my years spent at university.

Finally I would like to thank my friends who made my time here a more enjoyable experience.

Chapter 1

Introduction

1.1 The inverse problem of gravimetry

A classical problem in Newton's mechanics is the calculation of the gravitational potential Φ due to the mass density field ρ inside a compact domain Ω, which can be measured via the gravitational force $\vec{f} = \nabla\Phi$. Gauss's Law states that

$$\nabla \cdot \vec{f} = -4\pi\gamma\rho \tag{1.1}$$

where γ is the gravitational constant. Therefore Φ is also the solution to the Poisson equation

$$-\Delta\Phi = 4\pi\gamma\rho \tag{1.2}$$

in $\mathbb{R}^3$ with vanishing conditions at infinity.

Usually the domain Ω represents a roughly spherical object (for example the Earth) but we will be looking at a 3D cuboid representing a cargo container. Equation 1.2 describes the forward problem of calculating Φ based on the known density distribution ρ inside Ω, with the added assumption that $\rho = 0$ outside Ω. This is well-posed and the unique solution can be calculated using the technique of fundamental solutions [33]

$$\Phi(\mathbf{x}) = \int_\Omega k(\mathbf{x}-\mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \tag{1.3}$$

with the kernel

$$k(\mathbf{x}-\mathbf{y}) = \frac{\gamma}{|\mathbf{x}-\mathbf{y}|}. \tag{1.4}$$

In the classical inverse problem of gravimetry one tries to find $\rho(\mathbf{x})$ inside Ω from $\mathbf{g} = \nabla\Phi$ given either on (part of) $\partial\Omega$ or at a set of discrete points outside of Ω. For

example a sensor being put on planes flying over the area of interest in order to find sharp discontinuities indicating pockets of salt [48], [53] or oil. This problem was first analysed in Stokes in 1867 [77].

Existence and uniqueness of this inverse problem have been analysed in [44], [45], [58], [83] and [40] amongst others. In [45] it states there is an absence of existence theorems due to the range of operators for this problem not being closed in classical function spaces. It also makes the point that there are numerical difficulties arising from its stability under constraints is weak, resulting in very slow convergence for iterative algorithms and so a build up of numerical errors.

The same book makes the point that for $\nabla\Phi$ given on $\partial\Omega$ then, combined with the vanishing conditions at infinity, the exterior Dirichlet problem for Φ outside Ω can be solved. In fact Φ can be found uniquely outside Ω by the uniqueness in the Cauchy problem for harmonic functions. Thus we can find $\partial\Phi/\partial\nu$ on $\partial\Omega \in Lip$ where ν is the outward unit normal on $\partial\Omega$, so we have Cauchy data on $\partial\Omega$. As [40] puts it, the "Dirichlet to Neumann map" for this problem can be obtained from a single data measurement.

However this is not enough information to determine a unique solution. [83] contains a theorem which requires geometric restrictions to be placed on the unknown interior of Ω, with a similar theorem being stated in [44]. First restrict ourselves to finding an unknown object D within Ω. Let D be a compact region in $\mathbb{R}^3$ such that it is radially convex, which means that any straight line in $\mathbb{R}$ which passes through the center of mass of D will intersect the boundary ∂D at exactly two points.

Next we need to place certain restrictions on the density distribution ρ. We require it to be zero everywhere outside D and be a known non-negative density $\rho(\mathbf{r})$ (in polar coordinates) which is distributed spherically-symmetrically with respect to the center of mass. For example a special case would be where the density is a known constant ρ_0 everywhere within D. In this case the unknown object D can be uniquely recovered using the data recorded on $\partial\Omega$.

If the convexity condition is weakened then uniqueness begins to deteriorate. For the case where D is a spherical ball with known density the radius R (and therefore D) can be uniquely recovered. However if a smaller ball with the same center of mass is removed from D, creating an annulus of thickness $\hat{R} < R$ centered on the

center of mass, then uniqueness cannot be guaranteed. It is unlikely we will come across examples which fit these narrow requirements, and so non-uniqueness will be assumed.

Furthermore it turns out that the inverse problem is ill-posed in the sense of Hadamard violating all three conditions stated by Hadamard for a well-posed problem:

(i) In general there exists a solution only if Φ outside of Ω belongs to a certain function space of harmonic functions satisfying the Picard condition.

(ii) The Fredholm integral equation (1.3) has a quite large nontrivial null-space involving anharmonic functions such that in general the solution is non-unique.

(iii) Even when restricting the solution to a subspace orthogonal to the null-space of (1.3) the problem will be highly ill-posed.

A detailed discussion of these aspects of the inverse problem of gravimetry is given in [62] and [63] with the latter relating it to the case of a (Coulomb) potential V generated by an electric charge distribution.

While the discussion so far has highlighted the downsides of using gravimetry for imaging the density content of compact objects, this technology has gained extreme popularity in a variety of applications, in particular in geophysics and geodesy. [31] goes into detail about many of them, for use in navigation and exploration in the past, and [27] documents the increased use of gravity gradiometry. In such applications either no alternatives are available that show less serious difficulties, or it is used additionally to alternative techniques such as the imaging with electromagnetic or seismic waves - see [67], [54], [84] and [41].

1.1.1 The inverse problem of gravity gradiometry

Recent improvements in measurement technology have suggested a closely related way of obtaining gravity observations to be used for obtaining information on ρ. It turns out that so-called full-tensor gradiometers that measure the three cartesian partial derivatives of each of the components of $\mathbf{f} = \nabla\Phi$ are much less sensitive to certain types of environmental noise (such as vibrations or random accelerations of the carrying vehicles), with [9] comparing the improvement from the classical gravity gradiometers to those historically used in submarines. These can provide extremely accurate

measurements of a second order gradiometry tensor of the form

$$\mathbf{T} = \nabla\nabla\Phi = \begin{bmatrix} \left[\frac{\partial^2\Phi}{\partial x^2}\right] & \left[\frac{\partial^2\Phi}{\partial x\partial y}\right] & \left[\frac{\partial^2\Phi}{\partial x\partial z}\right] \\ \frac{\partial^2\Phi}{\partial x\partial y} & \left[\frac{\partial^2\Phi}{\partial y^2}\right] & \left[\frac{\partial^2\Phi}{\partial y\partial z}\right] \\ \frac{\partial^2\Phi}{\partial x\partial z} & \frac{\partial^2\Phi}{\partial y\partial z} & \frac{\partial^2\Phi}{\partial z^2} \end{bmatrix}. \tag{1.5}$$

See for example [19] and [30]. Since $\mathbf{T}$ is both symmetric and traceless, only five components are required, namely those in square brackets. The corresponding data can be described by a modification of (1.3) as

$$\mathbf{T}_{ij}(\mathbf{x}) = \int_\Omega K(\mathbf{x}-\mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \tag{1.6}$$

for $i = 1, 2$ and $j = 1, 2, 3$ with

$$K(\mathbf{x}-\mathbf{y}) = \gamma\frac{3(x^i-y^i)(x^j-y^j)-\delta_{ij}|\mathbf{x}-\mathbf{y}|^2}{|\mathbf{x}-\mathbf{y}|^5}, \tag{1.7}$$

where all coordinates are Cartesian and δ_{ij} denotes the Kronecker symbol ($\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise) [80]. Uniqueness of this modified inverse problem is discussed in [83]. It follows similar rules as discussed above for the gravimetry inverse problem. In particular, also $\mathbf{T}$ has a large nontrivial null-space and the corresponding inverse problem is highly ill-posed. As previously mentioned for gravity knowing $\mathbf{T}$ on the surface is equivalent to using Cauchy data. Its advantage lies in the mentioned stability with respect to certain types of noise along with an increased resolution over the standard gravimetry technique.

Gravity gradiometry has been in use in geophysical applications already for quite a long time for finding sharp discontinuities below the surface of the Earth [56]. Recently it has been proposed for the application of the screening of cargo containers for detecting concealed heavy fissile materials without the need of opening the containers in [51] which as also explored in an MSc dissertation, forming the beginnings of this study.

There are alternatives to consider other than using gravity gradiometry for imaging cargo containers. However due to the metallic shell usually encasing a cargo container there are problems when employing an electric or magnetic based imaging technique. [50] is a study performed on the effect of steel casing on electromagnetic data and finds there is increasing attenuation as the frequency increases, limiting the useable frequency to around 300 Hz and therefore limiting the resolution.

Imaging using gamma and x-rays has also been considered. However these run into practical limitations when considering potential damage to biological or electrical materials. In contrast gravity gradiometry imaging is non-destructive, as it measures what is already there without any need for an outside source.

A more promising technique is that of muon tomography, which has recently been studied with respect to imaging cargo containers in the last 10 years - see for example [32], [78], [11] and [6]. By measuring the muons detected from cosmic rays it results in a non-destructive scanning technique as it is measuring only what is already present. Muons are highly penetrative and so shielding of nuclear material would not present a problem.

The results look promising as [6] in particular contains details of a proof-of-concept type apparatus and algorithm combination. The accuracy of results are based on accuracy of muon momentum and time taken to detect muons. Using simulated data based on sensor performance at 50% accuracy a uranium block of size 100mm $\times$100mm $\times$100mm surrounded by scrap iron can be detected after 5 minutes. Using true momentum information it takes only 3 minutes.

In comparison gravity gradiometry has mainly been used in geophysical applications, which is on a length scale of several hundred meters. In fact to my knowledge after the initial paper [51] there does not seem to be further work performed for this application. The next mention of using gravity gradiometry to scan cargo containers specifically is a mention in [79], which cites the initial paper in a section detailing the applications of gravity gradiometry and others. It is usually applied to imaging underground, whether for oil, salt or water, on the larger length scales stated above. Using it on a comparatively smaller length scale such as a cargo container provides its own eccentricities compared to most of the geophysical applications studied previously.

It might be that both gravity gradiometry and muon scanning techniques have limitations, whether it be non-uniqueness for gravity or required scanning time for muons. A combination of the two approaches could prove fruitful in the future.

1.2 Discretised inverse problem of gravity gradiometry

For simulating an inverse problem of gravity gradiometry on a computer the forward and inverse problem need to be discretised first as is performed in [51]. This gives rise to a discretised inverse problem of the form

$$\mathbf{d} = \mathbf{G}\boldsymbol{\rho} \tag{1.8}$$

where $\mathbf{d}$ is the data vector indicating a discrete set of measurements of type (1.6), $\boldsymbol{\rho}$ is a discretized form of ρ following a specific discretisation scheme, and $\mathbf{G}$ is the corresponding sensitivity matrix representing a discrete form of the Fredholm integral operator of the first kind (1.5) in the chosen discretisation scheme.

Notice that in [83] it is indicated that the matrix $\mathbf{G}$ would have maximal rank, but that it is expected to be highly ill-conditioned reflecting the ill-posedness of the underlying continuous inverse problem. The proof is in Theorem 7 of that paper but it is mainly due to measurement errors. The null space would be the discrete version of the kernel of the forward map. Instead there is an "approximately null" subspace involving many eigenvalues relatively close to zero but none reaching it, resulting in a highly ill-conditioned matrix.

Generating data by the same discretisation method as used for the reconstruction comes with the risk of committing a so-called 'inverse crime' [25], [49]. In our proof-of-concept-style case study, we circumvent this risk by using a twice as fine discretisation level for generating the data than used when doing the reconstruction, giving rise to the search of a generalized solution of the form

$$\boldsymbol{\rho}^* = \operatorname{argmin}_{\boldsymbol{\rho}} \mathcal{J}(\boldsymbol{\rho}) \tag{1.9}$$

with

$$\mathcal{J}(\boldsymbol{\rho}) = \frac{1}{2}\|\mathbf{G}\boldsymbol{\rho} - \mathbf{d}\|_2^2 \tag{1.10}$$

and the assumption that

$$\mathbf{d} = \mathbf{G}\boldsymbol{\rho}_{true} + noise\,. \tag{1.11}$$

Additional regularization terms can be added to (1.9) as well. This problem has been addressed in [51] amongst others, which uses Tikhonov regularisation and a logarithmic barrier technique [55]. Other possible optimisation techniques could include total

variation regularisation [36]. In our work presented here we are instead using a model based approach where we use a level set representation of $\boldsymbol{\rho}$ which allows for discontinuities in $\boldsymbol{\rho}$ and at the same time adds regularisation to the problem. The final problem incorporates radial basis functions and applies a combination of steepest descent and genetic algorithm. Sources of note in these areas are [43], [53] and [59].

First to look at is [59]. This applies a local level set method for inverting gravity gradient data.. It describes the merits of incorporating a level set approach for gravity data - namely that we are dealing with closed irregular surfaces that are the boundary of an underlying domain D. A reliable and robust parameterisation for the surface is sought after, which is why the level set approach is chosen. It is able to change shape, along with merge and combine while it evolves, with no required input to do so.

There are differences in the methodology chosen. For instance [59] deals with a binary domain - looking for an unknown domain D of know density surrounded by zero density everywhere else. So only one level set is required but it also has to be initialised so that it encompasses a gravity center of a source. Therefore a weighted L_1 regularisation is performed to find an initial starting point for the level set function.

Moving on to [53] which applies the genetic algorithm to gravity data. As with the previous reference the domain is split into two (requiring only one level set function) where the density inside D is known and is zero elsewhere. The reason the genetic algorithm is used is to divide the domain into distinct values - this is due to the difficulties involved in applying derivative-base minimisation techniques to distinct regions. In comparison the use of a level set function creates a mapping from distinct values to a continuous function that techniques can work on.

The genetic algorithm employed in this paper is mostly similar to that used in this project. The only main difference is how much of the population is replaced at each generation. Here all but one are replaced, whereas [53] replaces the half of the population that fits the data the least. Both the genetic and steepest descent approaches have been explored in this project, culminating in a combination of the two.

Finally is the paper [43] which outlines the use of a colour level set approach and advocates its use in locating small objects in a domain. It is looking for tumours whereas we are looking for high density material but the aim is generally the same.

The data used in the paper is microwaves and so different choices had to be made. For instance the shapes of the domains and the structure within each domain is allowed to evolve. For gravity data only the shape is allowed to be known, the structure of the domain is assumed to be constant and known. Nevertheless this provided the basis for the colour level set approach used in this project, with differences in the interaction between level sets and the order of evolution. It also provided the idea of using a two-stage reconstruction - first find a density distribution that suitably matches the data without lead before using that as the starting point when looking for lead.

Notice that in our problem formulation the set of admissible solutions $\boldsymbol{\rho}$ are only required to satisfy

$$\|\mathbf{G}\boldsymbol{\rho}_{coarse} - \tilde{\mathbf{G}}\boldsymbol{\rho}_{fine}\| \leq \nu \tag{1.12}$$

where ν indicates the noise level of the data, $\tilde{\mathbf{G}}$ is the sensitivity matrix on the fine grid level acting on the fine grid representation $\boldsymbol{\rho}_{fine}$ and $\mathbf{G}$ is the sensitivity matrix on the coarse grid level acting on the coarse grid representation $\boldsymbol{\rho}_{coarse}$.

During this study it was assumed that the only noise added to the data was that generated by a difference in grid size when creating the data as opposed to reconstructing the density distribution. At one point the plan was to add noise depending on the possible gravity gradiometers that could be used, for instance [51] based the noise level on commercially available gravimeters. However this is designed primarily to be a feasibility study and so certain assumptions of ideal conditions had to be made in order to reach some conclusions. Therefore no background noise was added for any example shown.

The actual noise level can be difficult to define as it can depend on the unknown density within the cargo container. For some of the approaches it was assumed that as they slowed down or converged to a solution they had reached the full extent of their capabilities, and therefore found a suitable solution. However once the genetic algorithm was utilised it was more difficult to determine when it had sufficiently slowed down. So a more experimental approach was taken. This involved using various examples and comparing the data they made on the coarse grid to that on the fine grid. This gave us the expected noise level for that specific example. Then this had to be tied to an observable piece of data, which was the weight of the cargo container. With enough examples a more accurate noise level could be found.

This problem is still expected to contain a large number of admissible elements $\boldsymbol{\rho}_{coarse}$ that would qualify as potential solutions of the problem and that cannot simply be discarded. The underlying assumption is that an equivalent formulation holds when facing real data where then the fine grid level is replaced by a continuous level. In both cases (fine grid level and continuous level generated data) we again end up with an inherent non-uniqueness of the inverse problem that requires a special concept of solutions to our specific problem setup looking for potential threats hidden in the container. In particular, a single optimal answer satisfying a selected optimization criterion might not be the solution of choice, since a threat might be represented by a different member of the admissible set of solutions satisfying (1.12). The problem rather requires us to specify in our chosen model whether a threat is likely to be present or not, in a prescribed sense and considering more than one single member of the set of admissible solutions, perhaps a linear combination.

1.3 The structure of the sensitivity kernel G

An array of sensors will be used on the outer edge of the cargo container in the form of a sensor gate the cargo container passes through which can measure the gradient of gravity in all three coordinate directions. Using the coordinates portrayed in figure 1.1 the cargo container will move in the negative y-direction, along a conveyor belt type construction, thus creating an array of sensor locations on the top surface and the two long vertical sides. The gravitational potential of a point source is known to be

$$\phi = \frac{\gamma}{|\boldsymbol{r} - \boldsymbol{r_0}|} \tag{1.13}$$

where $\gamma = 6.67408 \times 10^{-11}\mathrm{m}^3\mathrm{kg}^{-1}\mathrm{s}^{-2}$ is the gravitational constant, $\boldsymbol{r}$ are the coordinates of the point source and $\boldsymbol{r_0}$ are the coordinates of the sensor.

Such measurements are stored in a vector $\mathbf{d}$. This style of presenting the gravitational potential is also present in [13]. [70] states that gravimeters can measure different combinations of these components, and looks into whether some measurements might be favourable for short distances. We simply regard all 5 components as equally valid, and so $\mathbf{d}$ will contain five pieces of data for each sensor.

Using more voxels in the finite element mesh gets us closer to the actual distribution

and increases the resolution. However it leads to a more under-determined system of equations, as we will most likely be physically limited by the number of sensor readings we can take, which may depend on the sizes of sensors created for this purpose and possible cost incurred. Also, increasing the number of voxels increases the size of the matrix **G**. There are several matrix-vector multiplications at each stage in the level set method that are unavoidable (due to the use of the forward problem at the very least) and so the computational time increases. Therefore a balance must be found.

The sensitivity kernel **G** stores the relationship between the density $\boldsymbol{\rho}$ and the observed data **d**. We begin by restricting our attention to a single sensor placed on the top surface of the cargo container exactly at the center of the rectangular edge. This provides five independent pieces of sensory data, each one a combination of all voxels within the cargo container. The contribution of each voxel to the sensory data is dependent on its relative location to the sensor, assuming all voxels are of equal size and mass. The resulting plots of these contributions give us a greater understanding of how each datum is influenced by the density distribution. Similar patterns can be found in [72] and [85].

It may be possible to plot the entire domain for each case, but it is largely unnecessary due to the sharp drop of in intensity of the gravity gradiometric data. Instead we have restricted ourselves to the most important parts of the cargo container. Figures 1.2 to 1.6 follow the coordinates shown in figure 1.1. Since the sensor lies directly in between two slices of voxels in the y-direction (and also the x-direction) it seemed necessary to include the cross-section of the cargo container one voxel either side of the sensor in the y-direction. These make up the first two plots in each figure, with the x and z coordinates increasing to the right and upwards respectively. The third plot is the cargo container when viewed from above with the y and x coordinates increasing to the right and downwards respectively.

Using these orientations we begin with figure 1.2, which is the component T_{xx}, relating to the second partial derivative of the gravitational potential in the x-direction. Perhaps this is best explained using the equation for T_{xx} as we find the 5 components analytically. If we allow the three components of distance to be

$$\bar{x} = x - x_0 \tag{1.14}$$

$$\bar{y} = y - y_0 \tag{1.15}$$

$$\bar{z} = z - z_0. \tag{1.16}$$

Then the first component can be written as

$$\frac{\partial^2 \phi}{\partial x^2} = \gamma \frac{2\bar{x}^2 - \bar{y}^2 - \bar{z}^2}{(\bar{x}^2 + \bar{y}^2 + \bar{z}^2)^{\frac{5}{2}}} \tag{1.17}$$

where γ is the gravitational constant. Due to the denominator this value is relatively negligible at a certain distance form the sensor. Restricting ourselves temporarily to the x-z planes at the top of the figure, this is where the distance in the y-direction is as small as it can be. Restricting ourselves further to the top row of voxels means that $\bar{z}$ is also relatively small, and so above a certain distance $\bar{x}$ in the x-direction from the sensor, the data would behave roughly like $|2/\bar{x}^3|$ and it does drastically increase along the top row towards the middle from either direction.

These non-negative values also extend slightly into the y and z-directions, creating faint spherical regions in the figures. As the distance $\bar{x}$ becomes relatively small this approximation breaks down, and the values for $\bar{y}$ and $\bar{z}$ take over to create negative values. Practically this should make sense, as the x-component of gravity should increase as distance decreases, but as soon as we reach a point under the sensor it fall dramatically (to zero or relatively close depending on measurement accuracy), resulting in a negative gradient component close to the sensors.

Figure 1.3 shows the same type of plots for the component T_{xy} which is

$$\frac{\partial^2 \phi}{\partial x \partial y} = \gamma \frac{3\bar{x}\bar{y}}{(\bar{x}^2 + \bar{y}^2 + \bar{z}^2)^{\frac{5}{2}}}. \tag{1.18}$$

since this involves the coordinate directions of x and y the most drastic change occurs in the x-y plane closest to the sensor, the lower plot. The placement of the sensor has split the domain up into four quadrants, which is the effect of separating the x and y-axis into two sections each. Let x^- denote the half of the domain in the negative direction from the sensor, which is the top half, and x^+ be the half in the positive direction, the lower half. Similarly let y^- and y^+ denote the left and right halves respectively. Then the sign of the component T_{xy} depends on what region we find ourselves in - if the signs of x and y agree then T_{xy} is positive and vice versa. This creates the vertical and horizontal discontinuities seen in the figure. Using a continuous domain instead of the mesh shown here would possibly smooth out the discontinuity. Also of note is that the maximum intensity occurs close to the sensor, but with a slight

distance before beginning to converge to zero in the middle. In fact the discontinuity is caused by a convergence to zero from different directions.

Moving onto figure 1.4 this involves the component T_{xz} calculated by

$$\frac{\partial^2 \phi}{\partial x \partial z} = \gamma \frac{3\bar{x}\bar{z}}{(\bar{x}^2 + \bar{y}^2 + \bar{z}^2)^{\frac{5}{2}}}. \tag{1.19}$$

The equation has the same format as 1.18, only with z instead of y, which should result in a similar result depending on x^+, x^-, z^+ and z^- as defined above. However since the tensor sits on top of the cargo container z^- represents the whole domain resulting in two distinct regions depending on x^+ and x^- only. Again agreement of the two has a positive T_{xz} and disagreement gives a negative T_{xz}. This makes sense as the z component of gravity would increase as we get close to the sensor and then rapidly decrease, so in the positive x-direction T_{xz} switches between positive to negative as it passes the sensor.

Next is figure 1.5 for T_{yy} which is

$$\frac{\partial^2 \phi}{\partial y^2} = \gamma \frac{2\bar{y}^2 - \bar{x}^2 - \bar{z}^2}{(\bar{x}^2 + \bar{y}^2 + \bar{z}^2)^{\frac{5}{2}}}. \tag{1.20}$$

Again this is similar to a previous component, only this time it is 1.17 as the $\bar{x}$ and $\bar{y}$ are simply swapped. The resulting plot is similar too - it has been rotated by 90° in the x-y plane, resulting in reflective symmetry in the y-direction.

Finally we come to figure 1.6 for T_{yz} which is calculated to be

$$\frac{\partial^2 \phi}{\partial y \partial z} = \gamma \frac{3\bar{y}\bar{z}}{(\bar{x}^2 + \bar{y}^2 + \bar{z}^2)^{\frac{5}{2}}}. \tag{1.21}$$

This has the same form as both 1.18 and 1.19 and most closely resembles figure 1.4. Both make use of the z-component and so both only have two distinct domains, this time the discontinuity occurs due to y^- and y^+.

The previous figures are useful in determining how one sensor is affected by the domain, but our problem will use many sensors combining their data together. To that end a test cargo container has been created where most of the domain has zero density (a vacuum) with only one small cube roughly in the middle of size 30cm along each dimension of density equal to 11g/cm^3 (a small piece of lead).

A sensor gate has been used to collect data on the three sides as described previously and then collected into the five distinct components to form figures 1.7 to 1.11.

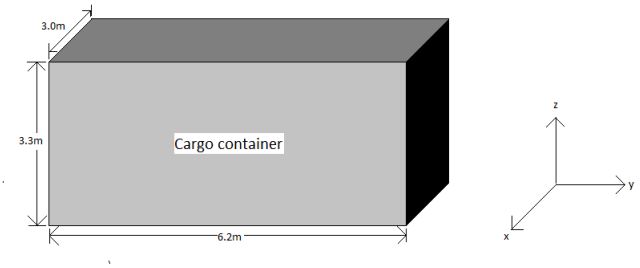

Figure 1.1: Dimensions of the cargo container.

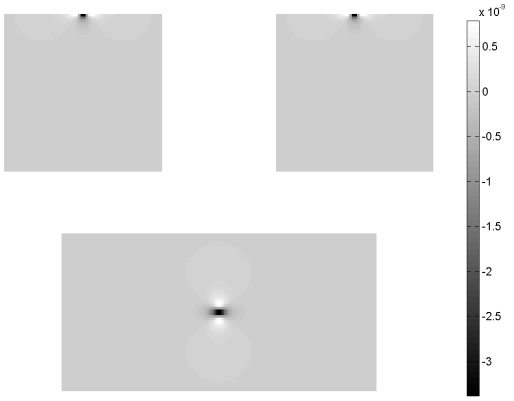

Figure 1.2: Plot of the cargo container for T_{xx} for a single sensor placed on top. Top two are vertical cross sections, bottom image is cargo container viewed from top.

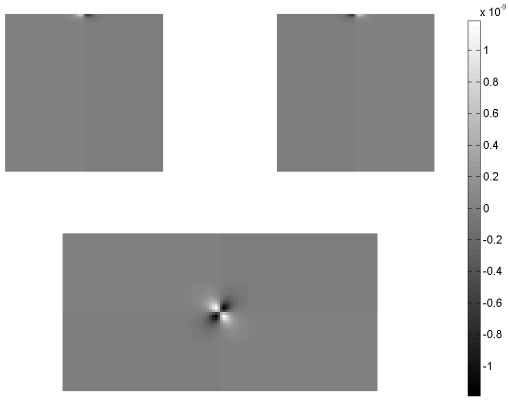

Figure 1.3: Plot of the cargo container for T_{xy} for a single sensor placed on top. Top two are vertical cross sections, bottom image is cargo container viewed from top.

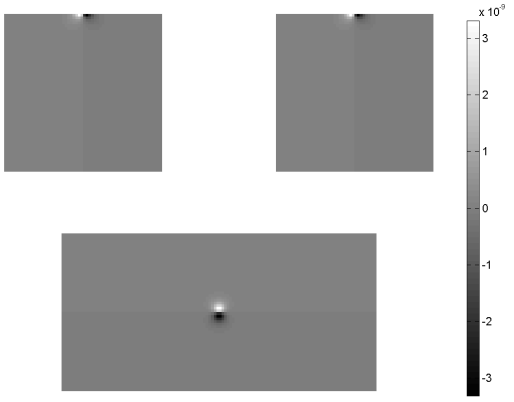

Figure 1.4: Plot of the cargo container for T_{xz} for a single sensor placed on top. Top two are vertical cross sections, bottom image is cargo container viewed from top.

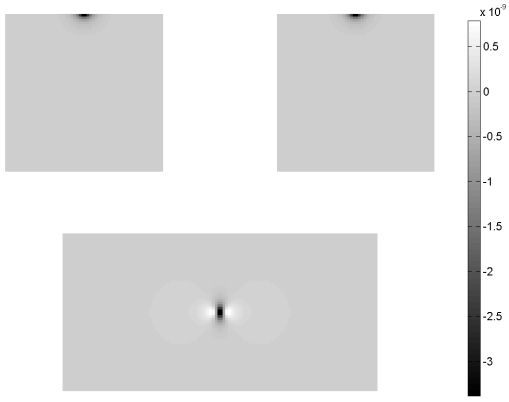

Figure 1.5: Plot of the cargo container for T_{yy} for a single sensor placed on top. Top two are vertical cross sections, bottom image is cargo container viewed from top.

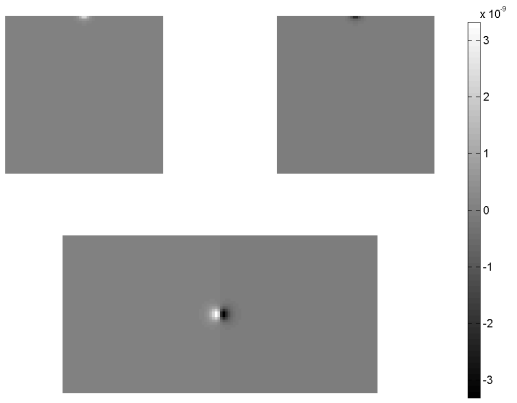

Figure 1.6: Plot of the cargo container for T_{yz} for a single sensor placed on top. Top two are vertical cross sections, bottom image is cargo container viewed from top.

This time each voxel represents a single sensor located at its center. The orientation of the figures emulates the cargo container being viewed from above, followed by the two long vertical sides being folded up to create a type of projection to the 2D plane. For all three plots the positive y-direction points towards the right. For the middle plot the value of x increases as we move down the page. Similarly the value of z for the top plot increases down the page whereas it increases up the page for the bottom plot.

Beginning with figure 1.7, which equates to T_{xx} everywhere, it has a similar structure to 1.2. In fact if we start with figure 1.2 and identify the small region of largest magnitude in the middle, then we expand it to the edges of the cargo container, it would resemble figure 1.7. The darker region has expanded across the top of the cargo container as this would be directly above the object, resulting in a lower value for $\bar{x}$ compared to $\bar{y}$ and $\bar{z}$, thereby making T_{xx} negative. However sensors on the sides would have relatively large values for $\bar{x}$, the largest of which would be in line with the object, resulting in the largest magnitude. In this case the object is closer to the vertical edge pictured at the bottom of the figure.

In figure 1.8 we see the equivalent plot of 1.3. The middle plot has recreated the pattern seen in figure 1.3, albeit with less distinct discontinuities. Instead there are bands along constant values for x and y, representing the location of the sensors in line with the object in the x and y-directions. This is a rough indication of the location of the object, approximately off center as shown in the middle plot. Also the distribution

further away is more spread out, reaching the edges of the cargo container.

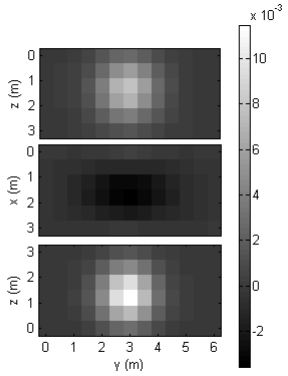

Figure 1.7: Plot of the cargo container for T_{xx} at the sensor locations when projected to a 2D plane.

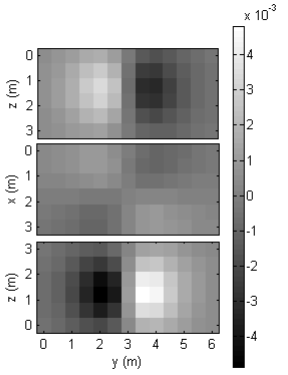

Figure 1.8: Plot of the cargo container for T_{xy} at the sensor locations when projected to a 2D plane.

Figure 1.9 is the case for T_{xz} and we see in the middle plot the same general pattern shown in figure 1.4 previously, only now with the added plots above and below it which show another band at a certain value of z as now the object has divided the domain into z^- and z^+. Therefore the domain is split into four regions.

The component T_{yy} is shown in figure 1.10 and, as before, it is a rotated version of 1.7. Since the cargo container is longer in the y-direction than in the x-direction it is slightly easier to see how much the distribution from figure 1.5 has been extended throughout the domain. Similarly figure 1.11 shows how the distribution from figure 1.6 has been extended and smoothed out to cover the whole domain, along with bands indicating the location of the object in the y and z-directions.

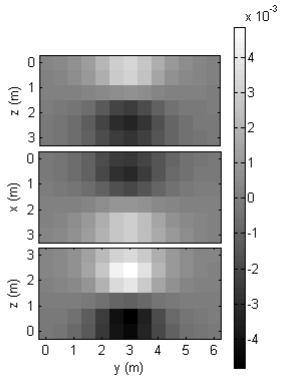

Figure 1.9: Plot of the cargo container for T_{xz} at the sensor locations when projected to a 2D plane.

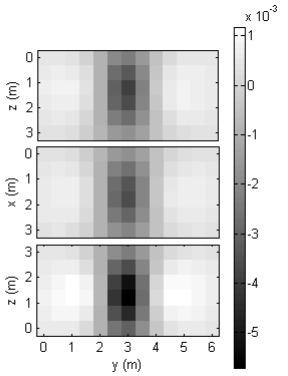

Figure 1.10: Plot of the cargo container for T_{yy} at the sensor locations when projected to a 2D plane.

Using only the above results we would be confident in predicting the location of an object of high density material within the cargo container if we already knew it was present. To be able to predict the presence of such an object we would have to compare the above data to that created when the object is absent. Given that the rest of the container is a vacuum removing the object will basically give us zeros for all pieces of data, which would indicate absolutely zero objects in the cargo container, but this case is entirely unrealistic. A more practical approach is to populate the background of the cargo container with objects of varying density such that no region reaches zero density (as air has a very low but non-zero density too). Below are two figures 1.12 and 1.13 created in the same way as before, both relating to the data component T_{xx} except that in figure 1.12 there is a lead object present which is absent in figure 1.13. There may be some tiny differences but to the human eye the two figures look identical, and so more accurate and reliable techniques need to be explored in order to determine the presence of lead or higher density material within a cargo container.

1.4 Building the matrix G

The sensitivity matrix $\mathbf{G}$ is an $5m \times n$ matrix where m is the number of sensors and n is the number of voxels. The way in which this was built was by using the analytical solutions formulated previously in equations 1.17 to 1.21. That is the formulation used for all examples seen in this project. It is possible errors were made when formulating this and so it was compared to an alternative approach, that of finite differences.

A central finite difference scheme was used on the gravitational potential Φ from 1.13, with the boundary condition $\nabla\Phi \cdot \mathbf{n} = \mathbf{0}$ on $\partial\Omega$, where $\mathbf{n}$ is the outward unit normal and $\partial\Omega$ is the boundary of the cargo container. From this we obtain an alternative sensitivity matrix that is applied to the density distribution ρ to make the data.

The two sensitivity matrices $\mathbf{G_a}$ and $\mathbf{G_d}$ created using the analytical solution and finite differences respectively were calculated using varying mesh sizes with a total of 96 sensors placed on the outside of the cargo container, at 0.1m distance from the surface. Then one was subtracted from the other component by component. The components of the resulting matrix were then squared and added together to find an

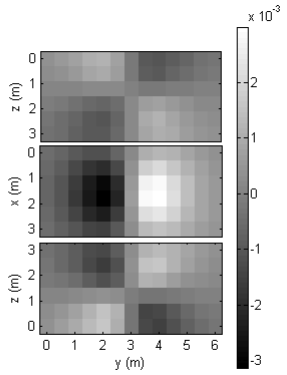

Figure 1.11: Plot of the cargo container for T_{yz} at the sensor locations when projected to a 2D plane.

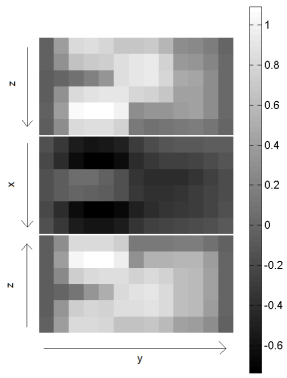

Figure 1.12: Plot of the cargo container for T_{xx} at the sensor locations for a more cluttered example.

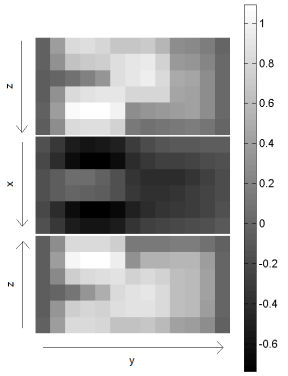

Figure 1.13: Plot of the cargo container for T_{xx} at the sensor locations for a more cluttered example without lead.

approximation to the error. This value is plotted in figure 1.14 against various voxel sizes, where the value on the x-axis represents the length of each edge of the voxel.

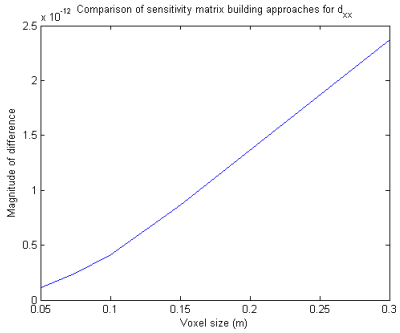

Figure 1.14: Comparison of sensitivity matrices built using analytical approach $\mathbf{G_a}$ and finite difference approach $\mathbf{G_d}$ for different voxel sizes. The difference of the two matrices was computed and then each component squared and added together to get an error approximation. Only the data involved in calculating d_{xx} is shown.

The idea behind this comparison is that both methods were computed separately. When compared if something does not make sense then one method must be wrong. If they seem to agree then it is unlikely that both will have the same sort of error, thus increasing the likelihood that the original method of building is correct. The finite difference scheme should become more accurate as the mesh becomes finer. As one can see from the figure the error decreases as the mesh becomes finer as expected.

In addition to this I performed some preliminary analysis on where the largest differences occurred. It was in close proximity to each individual sensor, which is unsurprising given the results seen later in this project - each sensor acts like a singularity and so errors can increase in magnitude drastically. Similar data was found when comparing the other 4 components of the tensor. It is therefore reasonable to assume that the method used to create the data throughout this project is accurate.

Chapter 2

Level set method with a gradient descent scheme

We have the cargo container as shown in figure 1.1 which we will denote with Ω. Our search is for a small region of high density material within Ω if there is such a region to be found. Denote this region $\mathcal{D} \subset \Omega$. The current unknowns are stored within the density distribution $\rho(\boldsymbol{x})$ for $\boldsymbol{x} \in \Omega$.

From here the logic is as follows. The most likely high density material is probably iron/steel, especially since a cargo container is usually made of weather resistant steel which is slightly below 8g/cm^3. Also copper and nickel are approximately 9 g/cm^3, which may be present in certain cargo containers. In contrast fissile materials have much higher densities. Uranium is around 19g/cm^3 and plutonium is around 16 to 19g/cm^3. A small amount of such materials would be dangerous, and it is unlikely we will be able to detect an object of that size. What we can assume is that there will be some shielding being utilised, such as lead which has a density of around 11g/cm^3.

For instance let us assume that a piece of fissile material that fills one voxel requires lead shielding of one voxel thickness in all directions (an assumption made by [51])R. This creates a cube of $3 \times 3 \times 3 = 27$ voxels, only one of which is fissile material. Assuming all of the fissile material is uranium means the average density of the cube is 11.3g/cm^3. So our aim would be to instead look for lead and assume there will be a sharp discontinuity between it and its surroundings.

Once we have reached above the density of copper, it becomes increasingly unlikely for a cargo container to be transporting objects of higher materials. Many such objects

are made of rare and precious metals such as silver (10.49g/cm^3), palladium (12g/cm^3), gold (19.3g/cm^3) and platinum (21.45g/cm^3) among others. These shouldn't show up very often, and would usually be on the cargo manifest when they do. Therefore it is a reasonable assumption to make that there will be a distinct jump between the density of lead and the surrounding objects. Using this assumption we begin by separating the domain into two distinct parts, with a known density in both,

$$\rho(\boldsymbol{x}) = \begin{cases} \rho_i & \text{for} \quad \boldsymbol{x} \in \mathcal{D} \\ \rho_e & \text{for} \quad \boldsymbol{x} \in \Omega \backslash \mathcal{D} \end{cases} . \tag{2.1}$$

Now the unknown is simply the domain $\mathcal{D}$. We introduce the function $\phi : \Omega \to \mathbb{R}$

$$\phi(\boldsymbol{x}) = \begin{cases} \phi(\boldsymbol{x}) \leq 0 & \text{for all } \boldsymbol{x} \in \mathcal{D} \\ \phi(\boldsymbol{x}) > 0 & \text{for all } \boldsymbol{x} \in \Omega \backslash \mathcal{D} \end{cases} \tag{2.2}$$

which is called the level set function corresponding to a domain $\mathcal{D}$ and has been used in papers such as [68], [73], [57], [29] and [46].

Our priority is finding $\mathcal{D}$, if it exists and so the structure of the region outside $\mathcal{D}^{(n)}$ is not as important, but that does not mean it can be neglected. It will contribute to the gravity readings and so some approximation must be used for this region. It may be that for certain cargo containers a single homogeneous density approximation of everything outside $\mathcal{D}^{(n)}$ is enough - namely cargo containers filled with a single type of item. However this may not always be the case.

It may be that this approach can be combined with another approach, such as x-rays to determine a starting point in such situations, and perhaps provide a useable ρ_i for other less homogeneous cases. Later on this model will be made more complex, but for now the aim is to find a sequence of functions $\phi^{(n)}$ which define domains $\mathcal{D}^{(n)}$ respectively such that $\mathcal{D}^{(n)} \to \mathcal{D}$.

As mentioned in the background to the uniqueness of the solution this separation into distinct regions is somewhat necessary. It would be very difficult to image the cargo container when both the shape and density of the objects are unknown. Therefore the density is assumed to be a known value.

2.1 Gradient direction

The idea behind a gradient descent direction is that we find a change $\delta\rho$ for $\boldsymbol{\rho}$ which would reduce the cost functional i.e. $\mathcal{J}(\rho+\delta\rho) < \mathcal{J}(\rho)$. The residual becomes

$$\mathcal{R}(\rho+\delta\rho) = \mathcal{R}(\rho) + G\delta\rho \tag{2.3}$$

where $\mathcal{R}'(\rho) = G$.

We define the inner product spaces

$$\langle p, q\rangle_P = \sum_{i=1}^{m} p_i q_i \quad \text{and} \quad \langle c, d\rangle_D = \sum_{i=1}^{n} c_i d_i \tag{2.4}$$

where m is the number of voxels and n is the number of data points.

Using 2.3 in (1.10) we obtain

$$\begin{aligned}
\mathcal{J}(\rho+\delta\rho) &= \frac{1}{2}\langle \mathcal{R}(\rho)+G\delta\rho, \mathcal{R}(\rho)+G\delta\rho\rangle_D \\
&= \frac{1}{2}\langle \mathcal{R}(\rho), \mathcal{R}(\rho)\rangle_D + \frac{1}{2}\langle \mathcal{R}(\rho), G\delta\rho\rangle_D + \frac{1}{2}\langle G\delta\rho, \mathcal{R}(\rho)\rangle_D + \frac{1}{2}\langle G\delta\rho, G\delta\rho\rangle_D \\
&= \mathcal{J}(\rho) + \langle \mathcal{R}(\rho), G\delta\rho\rangle_D + \frac{1}{2}\langle G\delta\rho, G\delta\rho\rangle_D \\
&= \mathcal{J}(\rho) + \left\langle G^T\mathcal{R}(\rho), \delta\rho\right\rangle_P + \frac{1}{2}\langle G\delta\rho, G\delta\rho\rangle_D
\end{aligned} \tag{2.5}$$

where $\mathcal{R}'(\rho)^* = G^T$ is known as the formal adjoint operator of $\mathcal{R}'(\rho)$.

If we were adjusting ρ then this would be straightforward, but we are actually adjusting the level set function. First let us imagine we have a level set function which defines for us a domain $\mathcal{D}$, and we use this to get the density distribution as defined in (2.1). Furthermore we now have the boundary of $\mathcal{D}$, $\partial\mathcal{D}$ which can be formally defined as

$$\partial\mathcal{D} = \{\boldsymbol{x} \in \Omega | \phi(\boldsymbol{x}) = 0\}, \tag{2.6}$$

or it can be more loosely defined as a narrow band of prescribed thickness where $\phi(\boldsymbol{x})$ changes sign, or where $\rho(\boldsymbol{x})$ has a sharp change in value. It can be enlarged by use of an extension velocity, such as those explored in [2] and [52].

An extension velocity is a way in which the evolution of the interface can be mapped to the rest of the domain, thus allowing the level set function to proceed faster. For example the level set function used in [1] uses the fast marching method ([75], [76]), which is a first order approximation for the velocity field near the interface.

Due to the strictly local solution for the characteristics near the surface this can lead to unexpected results. Therefore some work has been performed [24] in studying these characteristics and accounting for them.

Choose a point $\boldsymbol{x} \in \partial\mathcal{D}$. Now move it a small vector $\boldsymbol{y}(\boldsymbol{x})$ away from itself to create a new point $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{y}(\boldsymbol{x})$. Doing so for all points in $\mathcal{D}$ will create a new domain $\mathcal{D}'$ and boundary $\partial\mathcal{D}'$. This is shown in figure 2.1.

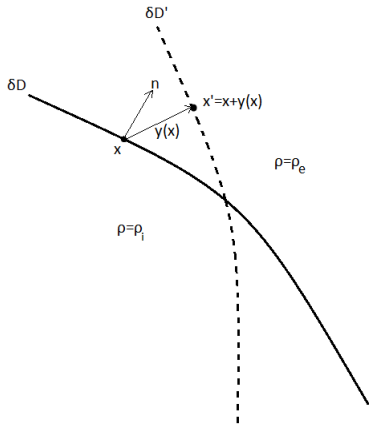

Figure 2.1: Deformation of the boundary of domain $\mathcal{D}$

In order to find $\delta\rho$ we can take the inner product with a test function f

$$\langle \delta\rho, f \rangle_{\Omega} = \int_{\Omega} \delta\rho(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = \int_{\text{symdiff}(\mathcal{D},\mathcal{D}')} \delta\rho(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \tag{2.7}$$

where $\text{symdiff}(\mathcal{D}, \mathcal{D}') = (\mathcal{D} \cup \mathcal{D}') \backslash (\mathcal{D} \cap \mathcal{D}')$ is the symmetric difference of the sets $\mathcal{D}$ and $\mathcal{D}'$ - the region which has changed from domain $\mathcal{D}$ into $\Omega \backslash \mathcal{D}$ or vice versa as this is the only region where $\delta\rho(\boldsymbol{x}) \neq 0$. This region is measure zero and so it can be approximated to a line integral

$$\langle \delta\rho, f \rangle_{\partial\mathcal{D}} = \int_{\partial\mathcal{D}} (\rho_i - \rho_e) \boldsymbol{y}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) f(\boldsymbol{x}) ds(\boldsymbol{x}) \tag{2.8}$$

where $\boldsymbol{n}(\boldsymbol{x})$ is the outward unit normal at $\boldsymbol{x}$ and $ds(\boldsymbol{x})$ is the incremental arc length.

There are two observations to be made of equation (2.8). First only the normal component of $\boldsymbol{y}(\boldsymbol{x})$ is required because if $\boldsymbol{y}(\boldsymbol{x})$ is perpendicular to $\boldsymbol{n}(\boldsymbol{x})$ then there would be no deformation, simply a rotation, and $\text{symdiff}(\mathcal{D}, \mathcal{D}')$ would be empty.

Secondly we have used the fact that $\delta\rho(\boldsymbol{x}) = \rho_i - \rho_e$ at the boundary point $\boldsymbol{x} \in \partial D$. Refer to figure 2.1. If the domain D expands outwardly, in the same direction of $\boldsymbol{n}(\boldsymbol{x})$

then a region which used to be equal to ρ_e is now ρ_i. Therefore the change is equal to $\rho_i - \rho_e$.

We can rewrite it in the form

$$\delta\rho(\boldsymbol{x}) = (\rho_i - \rho_e)\boldsymbol{y}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})\chi_{\partial\mathcal{D}}(\boldsymbol{x}) \tag{2.9}$$

where

$$\chi_{\partial D}(\boldsymbol{x}) = \begin{cases} 1 & \text{for } \boldsymbol{x} \in \partial\mathcal{D} \\ 0 & \text{for } \boldsymbol{x} \notin \partial\mathcal{D} \end{cases} \tag{2.10}$$

is known as the characteristic or indicator function.

Allow $\boldsymbol{y}$ to be both a function of $\boldsymbol{x}$ and of time t,

$$\boldsymbol{y} = \boldsymbol{v}(\boldsymbol{x})t \tag{2.11}$$

where $\boldsymbol{v}(\boldsymbol{x})$ is known as the velocity field. Using this in equation (2.9) and inserting it into equation (2.5) we get

$$\begin{aligned} \mathcal{J}(\rho(t)) - \mathcal{J}(\rho(0)) &= \langle \mathcal{R}'(\rho)^*\mathcal{R}(\rho), \delta\rho(\boldsymbol{x}, t)\rangle_P \\ &= \langle \mathcal{R}'(\rho)^*\mathcal{R}(\rho), (\rho_i - \rho_e)\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})t\chi_{\partial D}(\boldsymbol{x})\rangle_P \end{aligned} \tag{2.12}$$

or

$$\delta\mathcal{J} = \int_{\partial D} \mathcal{R}'(\rho)^*\mathcal{R}(\rho)(\rho_i - \rho_e)\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})ds(\boldsymbol{x}). \tag{2.13}$$

We simply need to find a function $F(\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})$ such that $\delta\mathcal{J} < 0$. Some alternative routes of calculating the gradient descent direction can be found in [59].

2.1.1 Method of steepest descent

Looking at (2.13) a possible choice is

$$F_{SD}(\boldsymbol{x}) = -(\rho_i - \rho_e)\mathcal{R}'(\rho(\boldsymbol{x}))^*\mathcal{R}(\rho(\boldsymbol{x})) \quad \text{for } \boldsymbol{x} \in \partial D \tag{2.14}$$

and this is known as the method of steepest descent. Substituting for $\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})$ in equation (2.13) we find it makes $\delta\mathcal{J}$ negative, thereby reducing the cost functional. The paper [17] looks into the functional analysis aspect of the method of steepest descent, along with the level set method.

To implement this we have to return to the idea of a level set function $\phi(\boldsymbol{x}, t)$ which varies over time. Restricting it to the boundary $\phi(\boldsymbol{x}, t) = 0$ and differentiating with respect to time will tell us how the boundary distorts

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \frac{\partial \boldsymbol{x}}{\partial t} = 0. \tag{2.15}$$

Next we need to recognise the forms of both velocity function

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{\partial \boldsymbol{x}}{\partial t} \tag{2.16}$$

and the unit normal to the boundary

$$\boldsymbol{n}(\boldsymbol{x}) = \frac{\nabla \phi}{|\nabla \phi|} \tag{2.17}$$

then we can rewrite (2.15) as

$$\frac{\partial \phi}{\partial t} + F_{SD}|\nabla \phi| = 0. \tag{2.18}$$

A level set function evolves over time so it makes sense to use an iterative formula. Start with an initial level set function $\phi^{(0)}$ and enact a finite difference scheme using a time step τ:

$$\frac{\phi^{(1)} - \phi^{(0)}}{\tau} = \delta\phi^{(0)} \tag{2.19}$$

where

$$\delta\phi = -F_{SD}|\nabla \phi|. \tag{2.20}$$

Continuing like this over successive time steps gives us the overall iterative formula

$$\phi^{(n+1)} = \phi^{(n)} + \tau\delta\phi^{(n)} \quad \text{for } \boldsymbol{x} \in \partial\mathcal{D}. \tag{2.21}$$

The level set method is only supposed to be enacted on the boundary of the domain $\mathcal{D}$. However this can be problematic as the boundary can represent a very small region in comparison to the whole domain, leading to very few changes at each iteration and so it could take a long time to converge to a solution. Also it can be difficult to define the boundary when using a finite element mesh since it is unlikely to find any voxels where $\phi(\boldsymbol{x}) = 0$ exactly. Instead we identify the surface at which $\phi(\boldsymbol{x})$ changes sign and use one voxel either side, thereby create a narrow band of voxels of thickness equal to two voxels.

An alternative sometimes used is

$$\partial D_\epsilon = \{\boldsymbol{x} \in \Omega | \, |\phi(\boldsymbol{x})| < \epsilon\} \tag{2.22}$$

for some $\epsilon > 0$. Increasing ϵ will increase the region on which we enact change, potentially speeding up the process. Also when we are restricted to the boundary of the level set and the domain $\mathcal{D}$ at the current iteration is relatively small, it can be difficult to recover an object further away. If the boundary region is extended, it becomes easier to recover such an object. The limits of ϵ would depend on the usual magnitude of $\phi(\boldsymbol{x})$ found near the boundary, which will change depending on location and time.

Many aspects of the model would have to be considered in order to use a value for ϵ with some certainty that it is valid. For instance the magnitude of the level set function will affect this, and the magnitude changes at each iteration of the method. If we want to we could remove all restrictions and allow the level set to deform everywhere in Ω. This would mean it is entirely plausible to recover an object anywhere in the domain in fewer steps. However there can be complications in this method as it reduces our control over the evolution of the level set function.

No value for τ has been specified so far in the iterative scheme. The steepest descent direction is calculated at the iteration n in order to find the level set at iteration $n+1$ and so, as is usual with finite difference schemes, the smaller τ is the more accurate to the actual steepest descent path the method will be. It will also take longer to converge and so larger τ values may have some merit, but they mustn't be too large or we run the risk of missing the minimum point.

τ should ideally be selected at each iteration and so some form of an adaptive method will be the most obvious choice. We want to choose τ such that $\mathcal{J}$ is reduced, but we don't want the domain $\mathcal{D}$ to deform too much after just one iteration. If there is drastic change it can be difficult to undo it if such change happens to be bad. The exact method used for finding τ will be explored before any results are shown or discussed. First some analysis on the model is required.

2.2 Smoothing the update

A level set function can define a unique domain $\mathcal{D}$, but a domain $\mathcal{D}$ can be defined using many different level set functions, approximately represented in figure 2.2. One level set function is steeper than the other, and so we have a chance to have a preference. A steeper level set function is in some ways more stable, as it takes larger magnitudes to increase/reduce it enough to change the domain. In contrast a smoother domain is more easily adjusted and so a deformation is more likely. We want the domain $\mathcal{D}$ to explore the cargo container before settling down, and so a smoother level set function is preferable. Depending on the line search method implemented it is possible for the update $\delta\phi$ to be too large, resulting in us jumping over the minimum. A smoother level set function is easier to adjust and correct the problem.

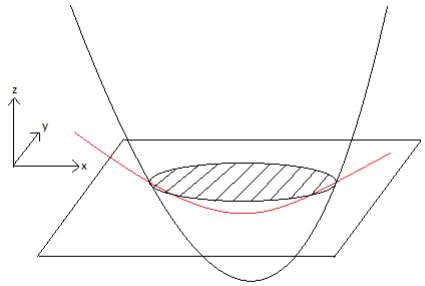

Figure 2.2: Two level set functions defining the same domain.

One way to attain this is by creating a new cost functional specifically for $\delta\phi$. Let $\psi_{dat} = \mathcal{R}'(\rho)^*\mathcal{R}(\rho)$ so that $\delta\phi = -(\rho_i - \rho_e)\psi_{dat}|\nabla\phi|$. Then a new cost functional is

$$\hat{\mathcal{J}}(\psi) = \frac{a}{2}\|\psi\|^2 + \frac{b}{2}\|\nabla\psi\|^2 + \frac{c}{2}\|\psi - \psi_{dat}\|^2. \tag{2.23}$$

ψ acts as a smoother alternative for the update $\delta\phi$. The use of it can depend on what we are attempting to model. For instance we might have highly oscillating boundary features for ϕ, leading to erratic features on the boundary. The use of ψ will smooth out the update $\delta\phi$ and the result is smoother and more regular boundary features. If the level set function has been initialised to form a spherical $\mathcal{D}$, then a smooth more spherical shape could be retained throughout. Without smoothing this spherical shape could deteriorate.

We are looking for a new ψ which minimises the above cost functional. There are three parameters a, b and c to be aware of. The first, a, denotes how much we would

like to minimise the magnitude of the update and b denotes how much we want to minimise the gradient of the update. c allows us to stay relatively close to the original update. The gradient direction is $((a+c)I - b\Delta)\psi - c\psi_{dat}$. At a minimum this would be equal to zero and so we can link the new update direction with the old using

$$((a+c)I - b\Delta)\psi = c\psi_{dat}. \tag{2.24}$$

Choosing $a = b = 0$ and $c = 1$ would simply give us $\psi = \psi_{dat}$. Going forward we fix $c = 1$ so that the magnitude of ψ stays relatively close to ψ_{dat}. Parameters a, b and c are not mutually independent, as it is their relative magnitudes that make the difference on the update. Also choose $b = \beta \geq 0$ and $a = \alpha - 1 \geq 0$ and we end up with

$$\psi = (\alpha I - \beta\Delta)^{-1}\psi_{dat} \tag{2.25}$$

which is accompanied by the boundary condition $\nabla\psi \cdot \boldsymbol{n} = 0$ on $\partial\Omega$, the boundary of Ω. In this reparameterisation α contains information on how much the magnitude of ψ should be minimised, thereby also dragging it away from ψ_{dat}. β controls the smoothness of ψ. However these could be combined further into $\gamma = \beta/\alpha$, resulting in a single smoothing parameter to find.

Alternatively we can find this via the inner product route. Let $\psi \in W_1(\Omega)$, where

$$W_1(\Omega) = \{\psi : \psi \in L_2(\Omega),\ \nabla\psi \in L_2(\Omega),\ \nabla\psi \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega\} \tag{2.26}$$

which is a Sobolev space [65]. Note that for $\phi \in W_1(\Omega)$ the trace of $\phi_{\partial D}$ (the zero level set) is only within the intermediate Sobolev space $W_{1/2}(\partial D)$ due to the trace theorem. Therefore the degree of smoothness of the reconstructed shape boundaries ∂D lies in between $L_2(\partial D)$ and $W_1(\partial D)$.

Next we use the weighted inner product norm

$$\langle u, v\rangle_{W_1(\Omega)} = \alpha \langle u, v\rangle_{L_2(\Omega)} + \beta \langle \nabla u, \nabla v\rangle_{L_2(\Omega)} \tag{2.27}$$

and then integrate by parts. This is easier to accomplish if we temporarily reduce to one dimension, with boundaries occurring at x_0 and x_l which represent the boundaries of Ω in one dimension.

$$\langle u, v \rangle_{W_1(\Omega)} = \alpha \int_{x_0}^{x_l} uv dx + \beta \int_{x_0}^{x_l} \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} dx \tag{2.28}$$

$$= \alpha \int_{x_0}^{x_l} uv dx + \beta \left[\frac{\partial u}{\partial x} v \right]_{x_0}^{x_l} - \beta \int_{x_0}^{x_l} \frac{\partial^2 u}{\partial x^2} v \tag{2.29}$$

$$= \int_{x_0}^{x_l} \left(\alpha - \beta \frac{\partial^2}{\partial x^2} \right) uv dx \tag{2.30}$$

where we have used $\frac{\partial u}{\partial x} = 0$ on $\partial\Omega$. This is the equivalent of $\nabla\psi \cdot \boldsymbol{n} = 0$ on $\partial\Omega$. From here it is a simple step to get to extend to 3 dimensions and since we have defined a value for c we are now left with two values α and β to define in order to solve

$$(\alpha I - \beta\Delta)\psi = \psi_{dat} \tag{2.31}$$

for ψ where ψ_{dat} is the direction of the update.

Some sources such as [38] put forward an approximation to this smoothing update, converting it to the heat equation using iterative time steps. This was explored to a certain extent but was not implemented for any later examples. The reason for an approximation can be due to speed, which can depend on factors such as the grid chosen. For the current state of the project it was deemed unnecessary, but it might become necessary if this approach is expanded upon for more complex grids. As such it is left as an avenue for further research.

2.3 Mesh choice

Before we begin with the level set method first we must decide on which finite element mesh to use. It is an important decision as it could potentially influence all future results. Assuming we have actual gravitational data any finite element mesh will be, at best, an approximation. Ideally we would choose the finest mesh possible, but there are other concerns, the first being the time required to create and use such a mesh.

Eventually the limitations may not be very strict in regards to computational power. However during analysis we will have to run the method many times and so any speed advantage is welcome. For our analysis we need a setup which can run for a relatively small timescale.

For both figures 2.3 and 2.4 a uniform mesh grid made up of voxel cubes was used. For whichever voxel size specified on the x-axis the cargo container was filled

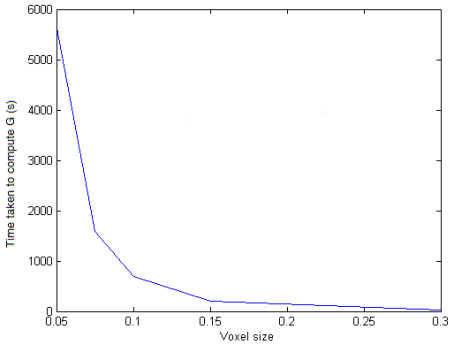

Figure 2.3: Time taken to set up the sensitivity matrices on varying voxel cube sizes. Number of voxels depend on the size of each voxel and range from 2000 to 432000.

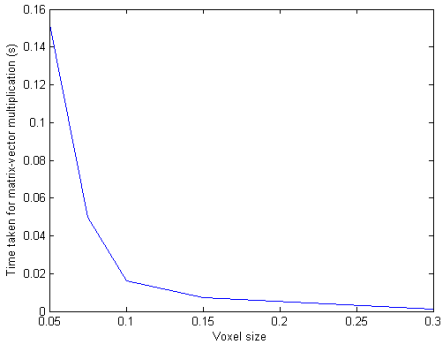

Figure 2.4: Time taken for a matrix-vector multiplication on varying voxel cube sizes. The matrix G is a $5m \times n$ matrix where m is the number of sensors and n is the number of voxels.

completely - bearing in mind the cargo container dimensions are 3m $\times$ 6m $\times$ 3m. So a voxel that is 0.3m on each side resulted in a grid of $10 \times 20 \times 10 = 2000$ voxels, whereas a voxel of length 0.05m resulted in a grid of $60 \times 120 \times 60 = 432000$ voxels. A random density distribution was generated with non-zero positive values to make sure the all voxels had a contribution to each sensor, and then this same density distribution was used for all calculations.

The matrix G is a $5m \times n$ matrix where m is the number of sensors and n is the number of voxels. For each sensor 5 rows are built based on the 5 independent tensor components. The analytical relations as shown in 1.17 and equivalent are used to populate the matrix, with the center of mass of each voxel used as the location of the gravitational point source and the sensor location also being assumed to be a point. Thus the matrix holds the contribution of each voxel to each piece of sensory data.

Figure 2.3 show the time taken to set up the initial information under different voxel sizes. Usually this would simply be the time taken to set up the sensitivity matrix G, which would be true in a practical scenario. However we do not have access to actual gravity data. Instead the usual practice under these conditions is to simulate such data using a finer mesh than that used for the reconstruction, thus avoiding the inverse crime. For each chosen mesh size, we have halved the length of each voxel and used that to create the data. For instance the data will be created using a mesh size of 0.15m if we need a reconstruction using voxels of size 0.3m. Therefore the set up includes the creation of the sensitivity matrix used in the level set method, and the

sensitivity matrix required to calculate the data any time we want to adjust the mesh size.

As one can see from figure 2.3 this time shoots sharply upwards as the mesh size is reduced. In fact there is a direct correlation. Halving the mesh size results in a cube one eighth the size of the original. Therefore we require 8 times the number of voxels and so it takes 8 times the time to set up. For instance using a mesh size of 0.1 requires approximately 700 seconds to set up, and a mesh size of 0.05 requires approximately 5600 seconds, which around 90 minutes (in this case there are 63 sensors, meaning 315 pieces of data).

Since it is part of the setup, this only has to be preformed once and so it might not seem very important for it to take 90 minutes. However this has an obvious effect to the inversion process when we need to enact the forward problem. Figure 2.4 shows the time taken for one matrix-vector multiplication under each mesh size. Using the iterative method outlined we will have to, at the very least, perform one matrix-vector multiplication at each iteration, with some methods requiring more than others, thus there is a growing effect on the time taken. Time is not necessarily the only factor when considering voxel size. It can also affect how well the density distribution fits the data. A smaller voxel size should logically lead to a better fit, but this may not always be the case. Tables 2.1 to 2.4 check some voxel sizes under a few different scenarios and find them lacking.

Sensor distance=1m. Object size=0.5m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1421	1539	92.3
0.2	1187	4732	25.1
0.15	3720	10693	34.8
0.14	10082	12636	79.8
0.13	12729	17200	74
0.12	11969	20286	59
0.11	20256	26450	76.6
0.1	37856	37856	100
0.09	34898	47824	73
0.08	38255	66309	57.7
0.075	80268	85544	93.8

Table 2.1: Number of locations within the domain where the actual cost functional is smaller than one for no object placed at that location, with sensor distance=1m and object size=0.5m.

Sensor distance=1m. Object size=0.6m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1421	1539	92.3
0.2	3118	4732	65.9
0.15	5225	10693	48.9
0.14	8291	10982	75.5
0.13	12079	15162	79.7
0.12	13886	20286	68.5
0.11	17460	23716	73.6
0.1	34375	34375	100
0.09	32551	43740	74.4
0.08	38653	61200	63.2
0.075	76035	79497	95.6

Table 2.2: Number of locations within the domain where the actual cost functional is smaller than one for no object placed at that location, with sensor distance=1m and object size=0.6m.

Sensor distance=1m. Object size=0.7m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	924	1152	80.2
0.2	1830	3888	47.1
0.15	6064	9216	65.8
0.14	8968	10982	81.7
0.13	10043	13284	75.6
0.12	12966	18000	72
0.11	16030	21168	75.7
0.1	31104	31104	100
0.09	30242	39884	75.8
0.08	44042	56347	78.2
0.075	65878	68231	96.6

Table 2.3: Number of locations within the domain where the actual cost functional is smaller than one for no object placed at that location, with sensor distance=1m and object size=0.7m.

Each of the four tables looks for an object of a different size. For each object size the voxel sizes are adjusted to see if it can be found. Such voxel sizes don't always divide into the object size, but that could add to the information gained. Similarly the voxel size doesn't always divide into the dimensions of the cargo container. In those situations as much of the cargo container is explored as possible. The object is then placed at all possible locations and its chances of being detected tested i.e. whether its presence fits the data better than the lack of an object.

Sensor distance=1m. Object size=0.8m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1088	1152	94
0.2	3216	3888	82.7
0.15	4489	7875	57
0.14	7903	9472	83.4
0.13	9318	11560	80.6
0.12	12030	15884	75.7
0.11	14587	18800	77.6
0.1	28037	28037	100
0.09	28039	36250	77.3
0.08	44248	51744	85.5
0.075	62040	63000	98.5

Table 2.4: Number of locations within the domain where the actual cost functional is smaller than one for no object placed at that location, with sensor distance=1m and object size=0.8m.

Voxel size for data	Factor of reconstruction voxels	% detectable
0.075	2	16.4
0.05	3	100
0.0375	4	16.4
0.03	5	100
0.025	6	16.4
0.01875	8	16.4
0.015	10	16.4

Table 2.5: Percentage of the domain where the object can be detected using different voxel sizes for the data.

Looking at tables 2.1 to 2.4 it is hard to discern a possible pattern. A voxel size of 0.1m will work for all these cases it seems, and 0.1 is a factor of all the object sizes. However there are voxel sizes which are also factors of some of the object sizes which do not achieve the same level of success. Therefore it cannot simply be a link between object and voxel size, there must be another factor. These are simple cases but further simplification is required in order to reach a conclusion.

We begin by fixing the voxel size for the reconstruction to be 0.15m. Then we place a single sensor on the top of the cargo container in line with the center of mass of a voxel $\boldsymbol{r_v}$. Following that we vary the voxel sizes when creating the data to compare the reconstruction to - call these the data voxels as opposed to the reconstruction voxels. The results are in table 2.5.

The voxel size chosen is 0.15m, with a finer mesh used to create the data. So far the data voxels have been half the length of the reconstruction voxels (one eighth the volume). Refer to table 2.5.

It seems that those data voxels whose length is an odd factor of the length of the reconstruction voxels give complete coverage, whereas those with an even factor do not. The results improve as we move the sensor away from the center of mass. So it seems to be an experimental error brought on by a mismatch between the data voxels and reconstruction voxels, and one we need to be very careful of. It is difficult to say for certain whether such an error will appear when using actual data. Based solely on what we have here it seems to be dependent on the data voxels, which don't exist in a practical scenario and so it's possible the error is not present in this case. However we don't have access to actual data and so we must find a way to remedy this error for the purposes of our analysis.

Although it is not included in table 2.5 this was also repeated using the same grid for the data as for the reconstruction (meaning the factor of reconstruction voxels would be 1). The object was found everywhere within the cargo container. This matches up with the trend seen in th table where it seems to work fine for odd factors but not even.

2.3.1 Distance between sensors and cargo container

In the previous section we looked at the placement of the sensors on the surface of the cargo container and found it to be tricky under most set ups to find an object in the reconstruction. This was due to a sensor acting like a singularity. Placing the sensor a certain distance from the surface should alleviate some of the effects of this.

The sensor grid is modelled as if each sensor exists at a point in space, essentially floating in a vacuum outside the cargo container as it is assumed there is zero density in between the sensors and the cargo container. The coordinates of each sensor point is used in the building of the sensitivity matrix G. Also the region extending from the sensor and beyond is assumed to be a vacuum. In reality there would be air, other sensors and many possible sources of interference. Here we focus instead on ideal conditions extending from the cargo container outwards.

Tables 2.6 to 2.9 are recreations of the setups shown in tables 2.1 to 2.4, only now

Sensor distance=1m. Distance from container=0.1m. Object=0.5m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1539	1539	100
0.2	4277	4732	90.4
0.15	10610	10693	99.2
0.14	12636	12636	100
0.13	17200	17200	100
0.12	20280	20286	99.97
0.11	26450	26450	100
0.1	37856	37856	100

Table 2.6: Percentage of locations within the domain where the object is detected, with sensor distance=1m, space between sensors and container=0.1m and object size=0.5m.

Sensor distance=1m. Distance from container=0.1m. Object=0.6m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1539	1539	100
0.2	4732	4732	100
0.15	10693	10693	100
0.14	10982	10982	100
0.13	15162	15162	100
0.12	20286	20286	100
0.11	23716	23716	100
0.1	34375	34375	100

Table 2.7: Percentage of locations within the domain where the object is detected, with sensor distance=1m, space between sensors and container=0.1m and object size=0.6m.

the sensors are placed a distance of 0.1m away from the cargo container. There is a noticeable improvement as most situations lead to complete coverage of the cargo container. Those that don't achieve complete coverage are using relatively large voxel sizes. Using the smaller voxel sizes of around 0.1m should be sufficient. From a practical standpoint it makes sense to have a buffer between the sensors and the cargo container as the container will be moving, increasing the risk of damage to the sensors. Due to the reduced penetration power of gravity gradiometry it would be preferable to place them as close as possible to the surface of the cargo container. After testing a range of values it was found that a distance of 0.1m from the surface of the cargo container is sufficient to achieve complete coverage of the domain when using most expected voxel sizes. An assumption has been made that this will be a sufficient distance to reduce the risk of damage to the sensors.

An interesting result is that of the first two lines in table 2.8 where the % useable

Sensor distance=1m. Distance from container=0.1m. Object=0.7m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	988	1152	85.8
0.2	3836	3888	98.7
0.15	9216	9216	100
0.14	10982	10982	100
0.13	13284	13284	100
0.12	18000	18000	100
0.11	21168	21168	100
0.1	31104	31104	100

Table 2.8: Percentage of locations within the domain where the object is detected, with sensor distance=1m, space between sensors and container=0.1m and object size=0.7m.

Sensor distance=1m. Distance from container=0.1m. Object=0.8m.			
Voxel size	Locations found	No. of possible locations	% useable
0.3	1152	1152	100
0.2	3888	3888	100
0.15	7875	7875	100
0.14	9472	9472	100
0.13	11560	11560	100
0.12	15884	15884	100
0.11	18800	18800	100
0.1	28037	28037	100

Table 2.9: Percentage of locations within the domain where the object is detected, with sensor distance=1m, space between sensors and container=0.1m and object size=0.8m.

is noticeably less than 100. This was most likely due to an inadequacy of the voxel size chosen and the approximation method followed. The reconstruction voxels were of size 0.3m which, when used to approximate an object of size 0.7m, actually only creates an object of size 0.6m in the reconstruction. In contrast the data voxels were of size 0.15m, which approximates the actual object with another object of size 0.75m. There is a noticeably difference between the two lengths in the reconstruction as opposed to the data voxels. Similarly using a reconstruction voxel of size 0.2 with data voxels of size 0.1m leads to an object of size 0.6m contrasting against an object of size 0.7.

This is also seen in table 2.6. Other tables will have mismatches but not to same extent as any difference in size is cubed when considering volumes as we are.

2.4 Line search method

In this section we will be focusing on the different aspects of the level set method described in previous sections, using results to explain choices throughout the method.

Back in section 2.1.1 it was mentioned that the method works only on the boundary of the object but that could be extended. It also warned of the reduction in control if taken too far - there is a limit to how far the boundary can be extended. For example refer to figure 2.5 which shows a cross section of the cargo container at 2.6m along the long edge. Here we have set up a case for the level set method using a voxel size 0.1m and a sensor spacing of 0.8m in relation to each other. No smoothing has been applied, and so $\alpha = 1$ and $\beta = 0$ (see section 2.2). The initial level set function is not important as we are mainly concerned with the regions of high $\delta\phi$, which in this case are those towards the edge of the container, at the positions of the sensors. Basically this is where the main update will occur and so we will most likely end up with objects clustered around the sensors in the reconstruction, missing the actual object.

Compare this to the case shown in figure 2.6, which is the same problem except $\delta\phi$ only acts on the boundary of the object. The initial level set ϕ is more clearly seen - it resembles a circle centered on the center of the cross section. As stated in section 2.1.1 the boundary is extended to either voxel around $\phi = 0$, effectively making a boundary of two voxels thickness, which is more clearly shown in this figure. In this case the method is working to reduce the size of the recovered object, and does not appear to be affected by the sensors at this time, giving us more control over the evolution of the level set. Therefore this approach is preferable.

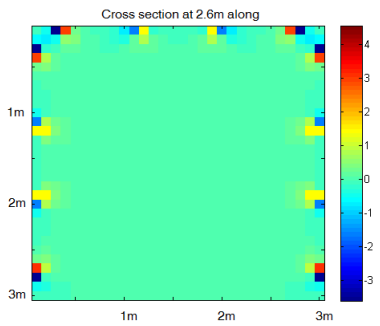

Figure 2.5: $\delta\phi$ is not restricted to δD, using $\alpha = 1$, $\beta = 0$.

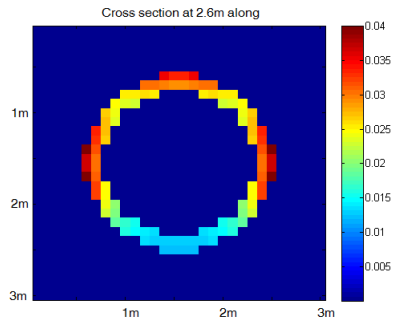

Figure 2.6: $\delta\phi$ is restricted to δD, using $\alpha = 1$, $\beta = 0$.

This does not preclude the use of some extension of the boundary in some future

work, as carefully chosen α and β can help matters. Refer to figure 2.7, where now β has been increased to $\beta = 2$, which guides $\delta\phi$ towards a smoother function, but with no alteration to the magnitude as $\alpha = 1$. Note how every point works to increase ϕ and so now the object could be less likely to end up near the sensors, but that also depends on subsequent iterations.

We must be careful as there are several choices to be made here. For instance the choice of the starting point can now influence where the method takes us. Also now that we have more control over the evolution of the level set it may be we are keeping it away from certain areas such as the sensors - the restriction leading to the other extreme by shifting the blind spots to the sensors. A balance must be achieved. Perhaps in the future a suitable extension velocity could be utilised in certain situations.

By adjusting both α and β we now have a wealth of opportunities to explore, but we have to be careful, as shown in figure 2.8. Here $\alpha = 2$ (with $\beta = 2$ remaining fixed), and now we find negative values of $\delta\phi$ in the corners of the cargo container, meaning objects are now drawn to those areas instead. Together these figures show that careful manipulation of α and β can improve our results, but can also lead to errors creeping in.

As a side note, both figures 2.7 and 2.8 have a slight change in $\delta\phi$ in an annulus of one voxel thick around the edge. This is due to the use of $|\nabla\phi|$ in $\delta\phi$. The function used is smooth up to the first derivative in the domain. However the boundary encoded in the central finite difference method used is $\nabla\phi \cdot \boldsymbol{n} = 0$, meaning that it is no longer smooth at the boundary, hence the sudden change.

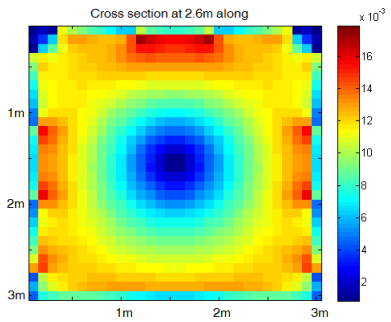

Figure 2.7: $\delta\phi$ is not restricted to δD, using $\alpha = 1$, $\beta = 2$.

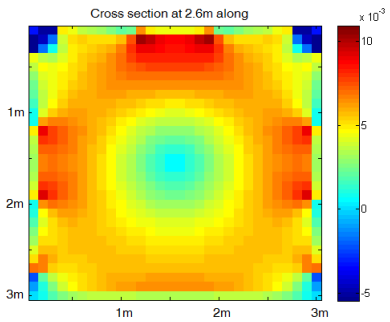

Figure 2.8: $\delta\phi$ is not restricted to δD, using $\alpha = 2$, $\beta = 2$.

Next we move onto the initial ϕ. It would be advantageous to initialise ϕ close to the actual solution, but that could result in an incorrect reconstruction if we begin the method in such a way. We don't know much about the inside of the cargo container, but hopefully it might be possible to learn something before we start.

The gradient of gravity can give us a good resolution of the object, but little to no penetration power. In contrast measuring the gravitational field gives us good penetration but not so much resolution. If we start with a sweep using the gravitational field it could provide us with a general region of where to look. Unless we start to introduce other detection techniques the only other measurement we can take is the weight of the cargo container, which could also provide us with useful information.

To illustrate the overall process of the evolution of the level set function refer to figure 2.9. Say some early results indicate the object is most likely somewhere in domain D_1. It might not be a circle (or a sphere in the case of the cargo container) but we can approximate it as one for now. So we make a starting point ϕ_1 which approximates the domain D_1.

Next we enact the level set method using a suitable line search. The evolution then usually fits into two stages. It is likely (and possible favourable) that we have overestimated the region in which the object lies, and so the first stage involves a reduction of the region, leading to a new domain such as D_k. If we are relatively accurate with our original assumptions D_k may be the actual object location, in which case the method stops. On the other hand, let D_n be the actual object location, and so we require D_k to move to D_n. There is a small region just outside the boundary of D_k on the side closest to D_n which therefore needs to change i.e. the function ϕ must switch from positive to negative in this region.

Given that in this case ϕ has switched from negative to positive already in stage 1, how positive it is depends on the line search method used in stage 1, or more precisely the magnitude of the step size τ. Sometimes it is tempting to use larger values for τ in stage 1 to minimise the cost functional more in a given steepest descent direction as this causes the object to change faster. However use of larger τ means larger gradients in ϕ so we may have to use even larger τ in stage 2 to adjust accordingly. Hence it is safer if we restrict τ to smaller values in stage 1.

Another problem occurs in stage 2 as the cost functional becomes flatter. For

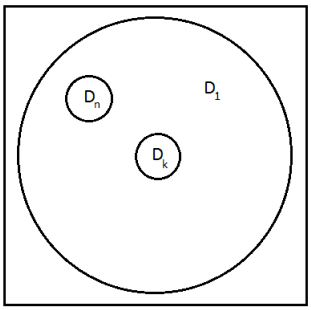

Figure 2.9: A simple representation of the changing of domains.

the moment let us instead look at a one dimensional domain defined by a level set function as shown in figure 2.10. The domain is the region where the level set function is negative. Let us consider the instance where the domain must move in the positive x-direction. If we are in stage 2 we can assume the domain is relatively small, usually small enough that it can all be considered to be on the boundary. So it is entirely possible for the object to disappear, as it has done in figure 2.11. At this point there is nowhere left for the method to go, and so it would halt.

In our experiments this usually corresponded to τ not being large enough. Instead using a larger τ can lead to something more like in figure 2.12. Here the original object has disappeared but a new object has formed too, in one jump from figure 2.10. Since the cost functional is flat there could be a range of values for τ which give us figure 2.11. Therefore during stage 2 larger τ were allowed to be used than in stage 1.

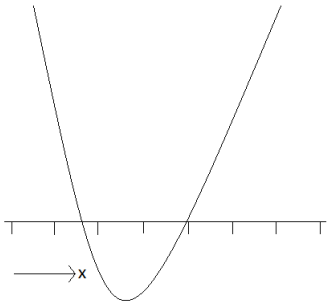

Figure 2.10: Starting point for a level set function.

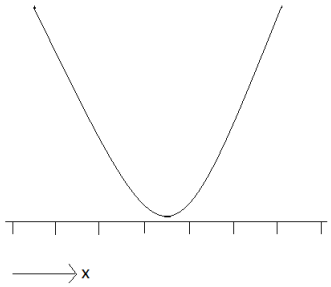

Figure 2.11: Possible level set function after k iterations.

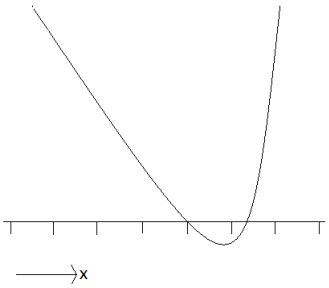

Figure 2.12: Possible level set function after n iterations.

As an example of the method working we have included figure 2.13, which represents a cross section of the cargo container. It is a simple example where the an

object of size one voxel and density 10g/cm^3 is placed near the edge of the container and surrounded by a constant density of 1g/cm^3, as shown in the top left figure. The method works in units of g/m^3, which is why the density is portrayed as a magnitude of 10^6. The second figure is the initial guess and the object evolves as we move along the top row, and carry on to the bottom.

There is no smoothing in this example. Stage 1 has been skipped over, which ends at 220 iterations, and took longer than expected as the size of τ was restricted to try and keep the magnitude of ϕ low. After this we can follow the movement of the object in the positive z-direction (using the same coordinates defined in figure 1.1). Stage 2 is relatively fast, only taking around 40 iterations during which the object creeps slowly towards the top edge of the cargo container. At it's conclusion the object resides relatively close to it's actual position. It never actually reaches the top edge, which is most likely a result of the singularity-type effects as we approach a sensor location.

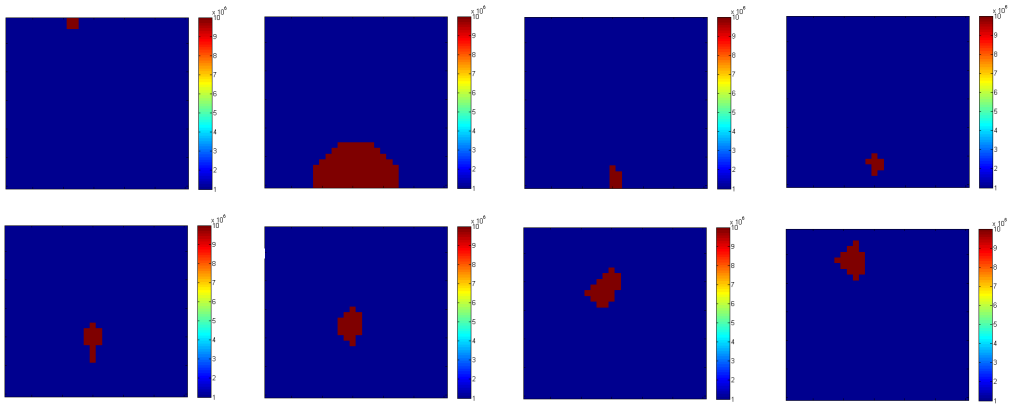

Figure 2.13: Level set method at various stages. Top-left is actual location. Second from left on top is the starting point and it continues right from there. Each image is a cross section of the cargo container taken at 3.4m along the longest length and each cross section is 3m $\times$3m.

At each iteration when the steepest descent direction $\delta\phi$ has been found, the line search implemented finds the approximate ideal step length in the following way, which requires an input of τ_{res} and initial τ_{max}.

Such a method is relatively simple and it increases in complexity in later sections but it suffices for our current requirements. τ_{res} is a value decided upon before, usually $\tau_{res} = 0.01$. If under a given τ_{max} the line search method simply returns $\tau = 0$ then τ_{max} is increased by a factor of 10, up to a proposed maximum of $\tau_{max} = 10^6$ (the

Algorithm 1 Find τ

```
1: procedure FIND τ
2:     while τ_max > τ_res do
3:         Calculate τ_int = τ_max/10.
4:         for i=1:11 do
5:             τ_i = (i − 1)τ_int
6:             φ_i = φ + τ_i δφ
7:             Calculate J_i using φ_i
8:         Choose τ = τ_i such that J_i is the minimum.
9:         if τ = 0 then τ_max = τ_int
10:    return τ
```

usual starting value is $\tau_{max} = 100$).

There are other checks within the method, for instance if the object is disappearing under a given τ then it instead looks for a τ corresponding to the second lowest cost functional. It might not be the most efficient way to do it but for now it provides useable results.

Figures 2.14 to 2.19 show a few more results. All contain a cube of size $2 \times 2 \times 2$ voxels, equating to a length of 0.2m on one side, but only one slice (one voxel thickness) was chosen in each case. Smoothing was used on some reconstructions but to keep results simple $\alpha = 1$ was used for all so the magnitude wasn't altered. The initial domain was the same for all results, comprising of spheres set in the middle of several slices similar to that seen in 2.6, but placed at certain intervals in the y-direction. Put together like this they comprised a tube running in the y-direction whose radius oscillates and has a maximum value of 1. In the following results this is used to simulate the effects of both not knowing where an object is and using a possibly wrong starting position.

Moving on to the results figure 2.14 shows an object close to the middle of slice 6, but slightly below and to the right. The reconstruction is shown in figure 2.15, where the object has been found in approximately the right location and it has the correct shape, so in this case the method has worked.

Compare that to the object shown in figure 2.16, which rests on the bottom of the cargo container, one of the furthest places away from the sensors as possible. The reconstruction, figure 2.17, does not quite manage to find the correct location of the object. This is not necessarily a bad result as it correctly identifies the actual presence

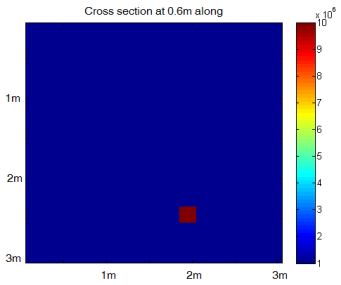

Figure 2.14: Actual location of the object in slice 6.

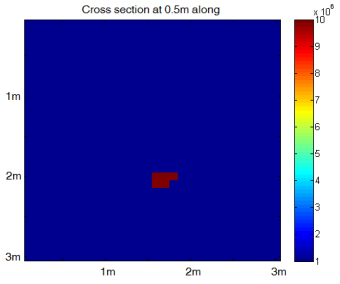

Figure 2.15: Reconstruction of 2.14 using $\beta = 0$.

of an object, and gets the distance in the y-direction correct.

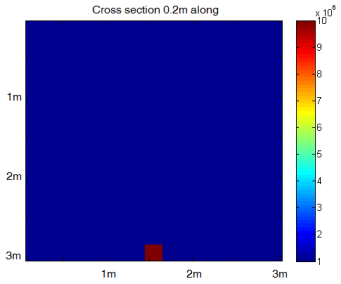

Figure 2.16: Actual location of the object in slice 20.

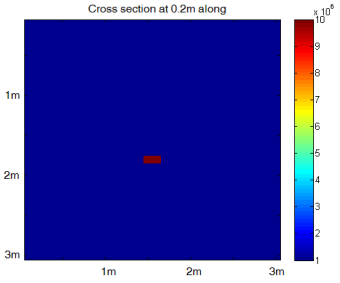

Figure 2.17: Reconstruction of 2.16 using $\beta = 0$.

The location can be improved by managing to place sensors on the bottom of the cargo container, as shown in figure 2.18. If that is not possible then simply using a different starting position can help, but then we could be using prior information we don't have access to.

Finally we move onto figure 2.19. This is working on the object shown at the start of figure 2.13 where the object is close to the sensors on the top of the cargo container. In that previous case the object was found rather easily, by use of a proper starting point for that example. In the current example something has gone wrong - not only is it stretched towards the middle, other objects remain in the reconstruction, which are not shown here.

Usually when the object starts to home in to it's actual position it doesn't leave anything at it's starting point. Unfortunately if during it's journey the object gets within a certain distance of the sensors, other regions of the container cease to change. The high sensitivity near the sensors overrides anything else. This is the reason to stay clear of the sensors for as long as possible.

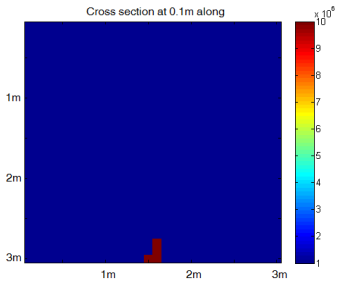

Figure 2.18: Reconstruction of 2.16 using $\beta = 0.003$ and sensors on bottom of cargo container too.

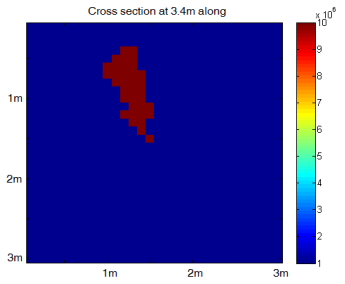

Figure 2.19: Reconstruction of 2.13 using $\beta = 0$.

As said above there are ways to ensure we get closer to the actual solution and so this is not much of a problem for the case of two distinct regions of the cargo container. However it will become more of a problem in a more complex case, when there will be regions of smaller densities possibly controlling the evolution of the rest if they get too close to a sensor.

This can lead to blind spots being generated that were not present at an earlier stage of the inversion process - certain objects masking others. It can get worse as the voxels become smaller in size as this allows them to get closer to the sensors - creating something more akin to a singularity. Some alternatives could be sought such as a non-uniform mesh - perhaps the voxels could be larger in close proximity to the sensors to keep the gravitational point source a certain distance away.

Something else worthy of mention is the lack of sensors around the domain in this case. It was shown that the inclusion of sensors on the bottom of the cargo container can improve the results for objects in that region. Alternatively we could simply add more to the 3 sides that currently have some. It could be that the localisation errors seen in the results are a result of the sensors. In which case it might not be simply the number of sensors but also the relative locations. A non-uniform sensor mesh may provide interesting results. This is an avenue for further study.

Now we have two parameters to define, but we can simplify it further. We can combine the two parameters into one, since what seems to matter more is the relative size of the parameters. Let $\gamma = \frac{\beta}{\alpha}$. Then instead we use

$$\psi = (I - \gamma\Delta)^{-1}\psi_{dat}. \tag{2.32}$$

As was stated previously there is some dependence between α and β, this simply takes

advantage of it to reduce the number of unknowns.

2.5 The usual colour level set

A simple level set method involves defining two domains by the use of one level set function. The colour level set is the method by which we define more domains using several level set functions. Let us extend to using two level set functions ϕ_1 and ϕ_2. Then the following domains can be defined

- $D_1 = \{\boldsymbol{x} : \phi_1 \leq 0 \text{ and } \phi_2 \leq 0\}$
- $D_2 = \{\boldsymbol{x} : \phi_1 \leq 0 \text{ and } \phi_2 > 0\}$
- $D_3 = \{\boldsymbol{x} : \phi_1 > 0 \text{ and } \phi_2 \leq 0\}$
- $D_4 = \{\boldsymbol{x} : \phi_1 > 0 \text{ and } \phi_2 > 0\}$.

Each domain D_i has an associated density ρ_i, which is assumed to be constant throughout. Next define the Heaviside function

$$H(\phi) = \begin{cases} 1 & \text{if} \quad \phi > 0 \\ 0 & \text{if} \quad \phi \leq 0 \end{cases} . \tag{2.33}$$

then the density distribution can be written as

$$\begin{aligned} \rho(\boldsymbol{x}) &= \rho_1(1 - H(\phi_1))(1 - H(\phi_2)) + \rho_2(1 - H(\phi_1))H(\phi_2) && (2.34) \\ &\quad +\rho_3 H(\phi_1)(1 - H(\phi_2)) + \rho_4 H(\phi_1)H(\phi_2). && (2.35) \end{aligned}$$

Now we can redefine the cost functional as

$$\mathcal{J}(\phi_1, \phi_2, \rho) = \frac{1}{2}||\mathcal{R}(\phi_1, \phi_2, \rho)||^2. \tag{2.36}$$

Our aim is to find forcing terms

$$f_1 = \frac{d\phi_1}{dt} \quad \text{and} \quad f_2 = \frac{d\phi_2}{dt} \tag{2.37}$$

such that $\mathcal{J}$ is reduced. Differentiating with respect to t gives us the partial derivatives

$$\frac{d\mathcal{J}}{dt} = \frac{d\mathcal{J}}{d\rho}\frac{\partial\rho}{\partial\phi_1}\frac{d\phi_1}{dt} + \frac{d\mathcal{J}}{d\rho}\frac{\partial\rho}{\partial\phi_2}\frac{d\phi_2}{dt}. \tag{2.38}$$

Using equation (2.35) and remembering that the differential of the Heaviside function is the delta function we find

$$\frac{\partial \rho}{\partial \phi_1} = \delta(\phi_1)[(\rho_3 - \rho_1)(1 - H(\phi_2)) + (\rho_4 - \rho_2)H(\phi_2)] \tag{2.39}$$

$$\frac{\partial \rho}{\partial \phi_2} = \delta(\phi_2)[(\rho_2 - \rho_1)(1 - H(\phi_1)) + (\rho_4 - \rho_3)H(\phi_1)]. \tag{2.40}$$

Proceeding using inner products as before we get

$$\frac{d\mathcal{J}}{dt} = \left\langle \mathcal{R}'(\rho)^*\mathcal{R}(\rho), \frac{\partial \rho}{\partial \phi_1} f_1 + \frac{\partial \rho}{\partial \phi_2} f_1 \right\rangle \tag{2.41}$$

and therefore the gradient directions are

$$f_1 = -(\mathcal{R}'(\rho)^*\mathcal{R}(\rho))\frac{\partial \rho(\phi_1, \phi_2)}{\partial \phi_1} \tag{2.42}$$

$$f_2 = -(\mathcal{R}'(\rho)^*\mathcal{R}(\rho))\frac{\partial \rho(\phi_1, \phi_2)}{\partial \phi_2}. \tag{2.43}$$

Due to ρ being dependent on ϕ_1 and ϕ_2 the act of changing one will have a knock-on effect for the other. Altering ϕ_1 would change ρ and therefore the residual and the gradient direction for ϕ_2. So one could formulate f_1 and then apply it, followed by a formulation of f_2 before applying it and repeating. Or one could formulate both and apply them both at the same time. This would be a slightly less strict steepest descent direction approach as a reduction for the cost functional in either one individually does not necessarily mean a reduction when both are applied at once. However it is faster to apply both and so this approach should not be easily dismissed.

An interesting aspect to this approach is that domains can disappear and reappear. For example, using the above define domains, it is possible for domain D_1 to disappear. However as long as both ϕ_1 and ϕ_2 create boundaries to work on, it is entirely possible to D_1 to return, but this won't be allowed to happen as it will reduce the control we have on the situation and attempting to identify allowable situations is complex and time consuming. We could go even further with the evolution. High density material is our focus and so we can estimate how dense it should be given a sufficient amount of lead shielding, which could be imposed upon the level set method easily.

2.6 Altered colour level set method

Anything of density lower than lead is unimportant to us and so we could attempt to approximate such density regions based off what we know of an average cargo

container. A good estimate for different domains could be

- D_1 for objects of density close to 0g/cm^3 (air/wood)
- D_2 for objects of density around 2g/cm^3 (plastics, some metals)
- D_3 for objects of density around 8g/cm^3 (iron, steel)
- D_4 for objects of density around 11g/cm^3 (lead).

These are simply rough estimates, for instance air and wood have been both added to domain D_1 because neither is important. The actual density of air is around 0.00125g/cm^3 and different types of wood have varying densities. If we are to assume the majority of wood found in cargo containers are the stacked pallets which are usually made of either oak or pine then the usual density is approximately 0.6g/cm^3. Therefore the above approximation should suffice.

[43] proposes an alteration to the general colour level set method, one which uses more than 2 level set functions for 4 domains. Applying this to the set up above we could define

- $D_1 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) \leq 0\}$
- $D_2 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) > 0 \quad \text{and} \quad \phi_2(\boldsymbol{x}) > 0 \quad \text{and} \quad \phi_3(\boldsymbol{x}) \leq 0\}$
- $D_3 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) > 0 \quad \text{and} \quad \phi_2(\boldsymbol{x}) > 0 \quad \text{and} \quad \phi_3(\boldsymbol{x}) > 0\}$
- $D_4 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) > 0 \quad \text{and} \quad \phi_2(\boldsymbol{x}) \leq 0\}$.

The reason for the use of 3 level set functions for 4 domains is to give us the freedom of splitting up the inversion process into two stages. During stage 1 we look for 3 domains $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ which requires the use of 2 level set functions. $\mathcal{D}_4$ is currently neglected. After stage 1 has finished converging we should be left with a possibly smoother (in a sense of maximum magnitudes rather that a smoothing of distinct domains) reconstruction of the cargo container. This can give us a suitable starting point for stage 2, thus avoiding some of the possible false results.

For instance it could be assumed that a high density object would present itself as part of domain $\mathcal{D}_3$ during the first stage, so the second stage could be initialised by portioning off part of domain $\mathcal{D}_3$ into $\mathcal{D}_4$. Then we could simply allow to method to

evolve to see how it reacts. It could attempt to remove $\mathcal{D}_4$ in which case there may not be any high density material. The aim is to reduce the number of unknowns further and attempt to get closer to an accurate solution. It showed good results in [43] for small tumours and so could be worth exploring for this application.

Using the same method as in the previous section we find the forcing terms f_i for the corresponding ϕ_i are:

$$f_i = -(\mathcal{R}'(\rho)^*\mathcal{R}(\rho))\frac{\partial \rho}{\partial \phi_i} \tag{2.44}$$

where the differentials are as follows:

$$\frac{\partial \rho}{\partial \phi_1} = \delta(\phi_1)\left[-\rho_1 + H(\phi_2)(\rho_2(1-H(\phi_3)) + \rho_3 H(\phi_3)) + \rho_4(1-H(\phi_2))\right] \tag{2.45}$$

$$\frac{\partial \rho}{\partial \phi_2} = \delta(\phi_2)H(\phi_1)\left[\rho_2(1-H(\phi_3)) + \rho_3 H(\phi_3) - \rho_4\right] \tag{2.46}$$

$$\frac{\partial \rho}{\partial \phi_3} = \delta(\phi_3)H(\phi_1)H(\phi_2)\left[\rho_3 - \rho_2\right]. \tag{2.47}$$

Using these forcing terms, along with an appropriate line search method, we can create an iterative method which provides us with a solution.

2.6.1 Inexact line search method

As discussed in a previous section, a proper line search method can be computationally expensive or, like the one used previously, relatively inefficient. An alternative to this is to used an inexact line search - one which doesn't necessarily find a minimum point in the decent direction, but a point which sufficiently reduces the cost functional. The Wolfe conditions are a tried and tested method of ensuring convergence to a minimum point.

We are trying to minimise the cost functional $\mathcal{J}(\rho)$, and so we calculate a descent direction $\delta\rho$. We need to find a step size τ such that $\mathcal{J}(\rho+\tau\delta\rho)$ satisfies the following conditions:

1.

$$\mathcal{J}(\rho+\tau\delta\rho) \leq \mathcal{J}(\rho) + c_1\tau\delta\rho^T\nabla\mathcal{J}(\rho) \tag{2.48}$$

2.

$$\delta\rho^T\nabla\mathcal{J}(\rho+\tau\delta\rho) \geq c_2\delta\rho^T\nabla\mathcal{J}(\rho) \tag{2.49}$$

for chosen parameters $c_1 \in (0,1)$ and $c_2 \in (c_1,1)$.

The first condition is also known as the Armijo condition and it finds a step length with a sufficient decrease in the cost functional. The right hand side is a straight line which includes the point at $\mathcal{J}(\rho)$ and has a negative slope $c_1\tau\delta\rho^T\nabla\mathcal{J}(\rho)$. Since $c_1 \in (0,1)$ (with [66] suggesting $c_1 = 10^{-4}$) there will be small step lengths which satisfy the condition, and in fact there will be an interval of step lengths $\tau \in (0,\tau_{max})$ which satisfy this condition.

However simply using small step lengths won't allow us to converge very quickly and this is where the second condition comes in, which is sometimes known as the curvature condition. The slope at $\mathcal{J}(\rho)$ in the direction of steepest descent is, by definition, negative. It should then increase as we move in that direction. If we were looking for a minimum it should eventually equal zero, but a sufficient increase will suffice.

The second condition can be adjusted to be

$$\left|\delta\rho^T\nabla\mathcal{J}(\rho+\tau\delta\rho)\right| \leq \left|c_2\delta\rho^T\nabla\mathcal{J}(\rho)\right| \tag{2.50}$$

to make the strong Wolfe conditions instead.

The method used to satisfy the Wolfe conditions is the same as the one found in [66], with some adjustments to it to make it suit our problem and its eccentricities but the pseudo code is still a good way to explain it, which is shown in algorithms 2 and 3, the second of which is a simple bisection method.

Algorithm 2 begins by accepting the range $(0,\tau_{max})$ where τ_{max} has been prescribed or calculated beforehand. Then the first τ_i is chosen in that range (usually in the middle). The first check is the first Wolfe condition. If this is not satisfied then the assumption is that τ_i is too large, and therefore an acceptable τ lies in the range (τ_{i-1},τ_i) and algorithm 3 is used to find it. If τ_i does manage to satisfy the first Wolfe condition then the second Wolfe condition is checked. Here it might not be necessary to use algorithm 3 as we may find an acceptable τ.

If the second condition is not satisfied then a final check is performed on the gradient of the cost functional at this step. A positive gradient implies we have moved past the minimum point and therefore have found a plausible interval in which τ may be found, in which case algorithm 3 is utilised. Otherwise we can come to the

conclusion that τ_i is too small and another is chosen in (τ_i, τ_{max}) before carrying onto the subsequent iterations.

Algorithm 2 Find τ

procedure FIND τ
  Set $\tau_0 = 0$, $\tau_{max} > 0$ and choose $\tau_1 \in (0, \tau_{max})$
  $i = 1$
  repeat
    Evaluate $\mathcal{J}(\tau_i)$;
    if $\mathcal{J}(\tau_i) > \mathcal{J}(0) + c_1\tau_i\delta\rho^T\nabla\mathcal{J}(0)$ **then**
      $\tau \leftarrow$ **bisect**(τ_{i-1}, τ_i) and **stop**
    Evaluate $\nabla\mathcal{J}(\tau_i)$
    if $\left|\delta\rho^T\nabla\mathcal{J}(\tau_i)\right| < \left|c_2\delta\rho^T\nabla\mathcal{J}(\rho)\right|$ **then**
      $\tau = \tau_i$ and **stop**
    if $\delta\rho^T\nabla\mathcal{J}(\tau_i) > 0$ **then**
      $\tau \leftarrow$ **bisect**(τ_i, τ_{i-1}) and **stop**
    Choose $\tau_{i+1} \in (\tau_i, \tau_{max})$;
    $i = i + 1$;
  until τ is found

Algorithm 3 is a simple bisection method that uses two main inputs τ_{lo} and τ_{hi} which correspond to the step lengths that create the lowest and highest cost functionals respectively which would be best explained by progressing through every eventuality, but is briefly described here. There are some alterations to this bisection method. It begins by finding the middle τ_i between τ_{lo} and τ_{hi} (the bisection part) and calculating the cost functional $\mathcal{J}(\tau_i)$.

The first check is the first Wolfe condition which, if it fails, will lead to further checking. For instance if $\mathcal{J}(\tau_i) > \mathcal{J}(\tau_{lo})$ then we have essentially improved on τ_{hi}, and therefore $\tau_{hi} = \tau_i$ and we return to the bisection step.

Alternatively if $\mathcal{J}(\tau_i) < \mathcal{J}(\tau_{lo})$ then we have improved on the τ_{lo} and so τ_i will replace it. However we must make sure we keep the minimum between τ_{lo} and τ_{hi}. If the gradients at them are different signs then we must also assign $\tau_{hi} = \tau_{lo}$ before assigning $\tau_{lo} = \tau_i$ and then move back to the bisection step.

If it did satisfy the first Wolfe condition then the second is check. Assuming W2 is not satisfied we must make sure the minimum is kept between the two approximations before returning to the bisection step.

Algorithm 3 bisect

procedure BISECTION METHOD
  repeat
    Set $\tau_j = |\tau_{lo} - \tau_{hi}| /2$
    Evaluate $\mathcal{J}(\tau_i)$;
    if $\mathcal{J}(\tau_j) > \mathcal{J}(0) + c_1\tau_j\delta\rho^T\nabla\mathcal{J}(0)$ **then**
      if $\mathcal{J}(\tau_j) > \mathcal{J}(\tau_{lo})$ **then**
        $\tau_{hi} \leftarrow \tau_j$;
      else
        if $\delta\rho^T\nabla\mathcal{J}(\tau_j)\delta\rho^T\nabla\mathcal{J}(\tau_{lo}) < 0$ **then**
          $\tau_{hi} = \tau_{lo}$
        $\tau_{lo} = \tau_i$
    else
      Evaluate $\nabla\mathcal{J}(\tau_j)$
      if $\left|\delta\rho^T\nabla\mathcal{J}(\tau_j)\right| \leq \left|c_2\delta\rho^T\nabla\mathcal{J}(0)\right|$ **then**
        $\tau = \tau_j$ and **stop**;
      if $\nabla\mathcal{J}(\tau_j)(\tau_{hi} - \tau_{lo})$ **then**
        $\tau_{hi} \leftarrow \tau_{lo}$
      $\tau_{lo} \leftarrow \tau_j$;
  until τ is found

2.6.2 Gradient descent results

Below in figures 2.20 and 2.21 we see two cross sections of a cargo container with a few different densities within. Figure 2.20 shows a part of the cargo container which has a lead box surrounded by an object mostly made of wood. The lead box also contains a small piece of plutonium not included that in the figure in order to keep the scale constant for all. Figure 2.21 shows another part of the cargo container where a large metal object has been placed. Of course an actual cargo container will have a much more complicated and mixed cargo container but this is where we will begin.

Reconstructing the density distribution falls into two stages. The first stage involves only changing level set functions ϕ_1 and ϕ_3, leaving $\phi_2 > 0$ for the entire domain. Thus there is no lead reconstructed in the first stage. For this example the starting cost functional was $\mathcal{J} = 41.45$, which is relatively far from being a good starting point. We can see from figure 2.22 that after stage 1 the lead box and wood mixture has been replaced with a metal and wood mixture, the metal covering a larger region than the lead. Looking at figure 2.23 we can also see that the method has reconstructed the large metal region quite accurately, although that is most likely helped by the size of the metal object. At this stage the cost functional was of the order 10^{-1}.

However our main focus is to identify lead and above. To do this we first adjust ϕ_2 so that a region of lead appears in the reconstruction. This not only requires to make ϕ_2 negative, it needs to be made negative in a region where $\phi_1 > 0$. In this case it has been simply prescribed it to the middle of the cargo container, and then allowed it to evolve, and from this a region of lead has been reconstructed in figure 2.24. It is not exactly the correct size and location but its presence is enough. Also some metal has been falsely reconstructed, but we aren't interested in anything but lead. The state of the other metal region after stage 2 is shown in figure 2.25.

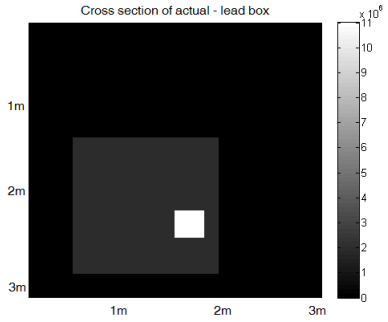

Figure 2.20: Actual density distribution in the region containing the lead box.

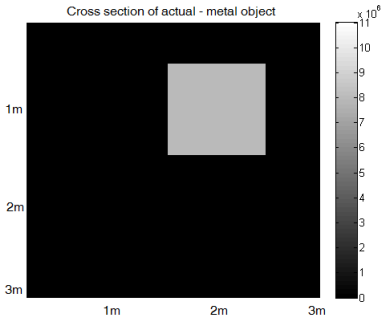

Figure 2.21: Actual density distribution in the region containing the metal object.

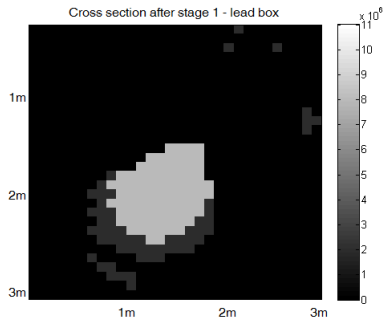

Figure 2.22: Reconstruction of the region containing the lead box after stage 1.

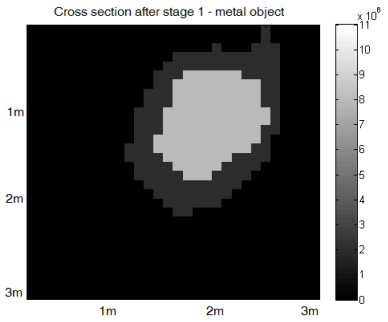

Figure 2.23: Reconstruction of the region containing the metal object after stage 1.

However there was one condition that had to fulfilled to get to this reconstruction. Stage 2 was first attempted by placing a lead object in the middle of the cargo container of only a few voxels in size to see what it would do to the method. Unfortunately instead of the other regions changing to accommodate it's presence, the method simply worked to remove the lead from the reconstruction. The code created would not allow this to happen, as we require (and prefer) at least one voxel of lead at all times.

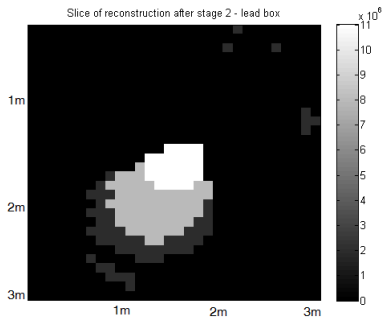

Figure 2.24: Reconstruction of the region containing the lead box after stage 2.

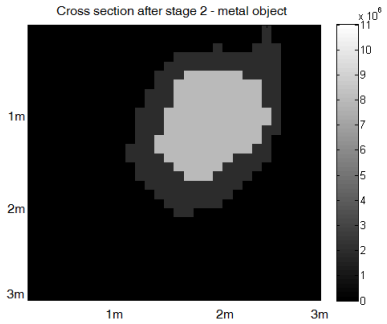

Figure 2.25: Reconstruction of the region containing the metal object after stage 2.

The other regions would not change and so the method essentially found a local minimum point. Although another possibility is the cost functional becoming flat. The way in which the reconstructions above were obtained was to increase the size of the lead object at the start of stage 2. At a certain size it creates enough of an impact to cause change in the other domains, and we eventually end up with the reconstructions above. The cost functional is not reduced below a magnitude of 10^{-1} in this example, but that is roughly the cost functional a correct solution should have. We have managed to reach the region of a feasible solution, except that we have had to force it along to get there.

Most of the effort so far has been devoted to finding a positive result, but we must also look at the possibility of false positives. For that reason the method was ran with a similar actual density distribution, the only difference being the absence of any lead or plutonium, and a very similar result to that above was found. It seems that, for this case at least, when we look for lead we find it, whether it is actually there or not.

The choices made during the algorithm have guided it to a set of solutions which, although having a low enough cost functional to be considered in the null space, will construct lead of the same size and shape whether it is actually present or not. Too many false positives mean this is not a viable option.

At this point it is worth mentioning how the results so far have been assessed in terms of quality. Throughout this project an undecided element was the process of processing certain results - for instance whether or not a trained observer would be required or whether hard data would give us a solution. This was left as an open possibility for a while and so these early results were judged on a qualitative measure

rather than quantitative for a while, with the reduction in the cost functional mainly begin used as a guide for convergence. However this approach can only take us so far and so later on more quantitative results are presented instead.

Chapter 3

Genetic algorithm working on the colour level set method with radial basis functions

3.1 Voxel-based genetic algorithm

Another avenue of enquiry would be the use of a genetic algorithm, one such method being known as PIKAIA, as found in [22]. The more general overview of such problems can be found in [37], whereas the paper [53] uses it on gravity data. Also it takes up a chapter in [74] which also includes other more statistical methods involving geophysical problems. The idea is the use the method of inheritance seen in nature, thereby allowing new generations to be "fitter" (i.e. fit the data better) than their parents. Over successive iterations the population should converge to individuals that fit the data to an appropriate degree.

The general steps of the genetic algorithm are as follows, with each being expanded upon after.

1. Define an individual I to be a density distribution $\boldsymbol{\rho}$ on the domain $\mathcal{D}$.
2. Encode $\boldsymbol{\rho}$ into a string of digits from 0 to 9 and call it a chromosome c_i relating to individual I_i.
3. Repeat part 2 for many more individuals to create a population of individuals I_i $i = 1, \cdots, n$, each with their corresponding c_i and $\boldsymbol{\rho}_i$ respectively.

4. For each $\boldsymbol{\rho}_i$ find the corresponding cost functional $\mathcal{J}_i$ and rank them in order of fitness - the lowest $\mathcal{J}_i$ will have rank $r_i = 1$ and so on.

5. Randomly choose 2 individuals using a probability depending on their fitness - those that fit the data better (i.e. lower cost functional) being more likely to be chosen.

6. A "crossover" event occurs wherein the chromosomes c_i and c_j combine information to create 2 offspring chromosomes $\hat{c}_i$ and $\hat{c}_j$.

7. Continue until there is a population of n offspring, then have the offspring replace the parents.

8. For each offspring each digit within its chromosome has a small chance of mutation - a random change from one digit to another to retain some semblance of genetic diversity.

9. Decode each offspring chromosome $\hat{c}_i$ the corresponding $\boldsymbol{\rho}_i$ and $\mathcal{J}_i$. Then repeat from 5) onwards until convergence occurs.

Let us look at the pixel-based reconstruction where the domain is split up into a finite element mesh, creating nv voxels in three dimensions. Starting off with a random density distribution, each voxel will have a value assigned to it. The first step is to encode this information into a chromosome-like structure. For the purposes of encoding we need all such values to lie in the interval $(0, 1)$, and so, like in the Tikhonov method in [51], a maximum and minimum value must be assigned. Realistically the minimum density would be $\rho_{min} = 0$ and, since the maximum density we look for is either plutonium or lead, we have an obvious choice of ρ_{max} being either of these two values. Dividing each voxel by ρ_{max} leaves us with the interval $(0, 1)$.

Next we need to decide on how much precision we require. A higher precision means more accurate results, but it also leads to an increase in computational time as each digit is another unknown to be found. Thus, when nd =number of decimal

places stored is decided upon, we have a modified density distribution.

$$\hat{\boldsymbol{\rho}} = \begin{pmatrix} 0.\hat{\rho}_{1,1}\ \hat{\rho}_{1,2}\ \cdots\ \hat{\rho}_{1,nd} \\ 0.\hat{\rho}_{2,1}\ \hat{\rho}_{2,2}\ \cdots\ \hat{\rho}_{2,nd} \\ \vdots \\ 0.\hat{\rho}_{nv,1}\ \hat{\rho}_{nv,2}\ \cdots\ \hat{\rho}_{nv,nd} \end{pmatrix}, \tag{3.1}$$

which has to be encoded into a chromosome string and so each component of $\hat{\boldsymbol{\rho}}$ is stored as a sequence of integers, representing the digits after the decimal point. Then they are concatenated into

$$\text{chromosome} = \hat{\rho}_{1,1}\ \hat{\rho}_{1,2}\ \cdots\ \hat{\rho}_{1,nd}\ \hat{\rho}_{2,1}\ \hat{\rho}_{2,2}\ \cdots\ \hat{\rho}_{nv,nd} \tag{3.2}$$

using the encoding technique. One chromosome represents a single individual within the population. In order to correctly mimic the inheritance procedure we need a population to work on, np =number of individuals in the population. The crossover stage is then performed in a few steps.

First, different members of the population will fit the data better than others and the cost functional can be used to measure the success. For each individual i, we find a function for the fitness S_i of the individual - the greater the fitness the larger S_i. So far the cost functional $\mathcal{J}$ has been used as an error measure which in a sense operates in the reverse to fitness - aiming for a lower $\mathcal{J}_i$ as opposed to a larger S_i. A suitable conversion could be incorporated or an alternative fitness measure found.

The choice of how to calculate the fitness has to be made carefully, as it can have drastic effects on the results. To explain the possible effects the selection process needs to be explained. For now the fitness is left as an abstract measure.

Calculate the sum of all fitness values

$$F = \sum_{i=1}^{np} S_i \tag{3.3}$$

and define a running sum

$$T_j = \sum_{i=1}^{j} S_i \quad , \quad j = 1, \ldots, np. \tag{3.4}$$

From this $T_{j+1} \geq T_j\ \forall j$ and $T_{np} = F$. Next generate a random number $R \in [0, F]$, and find the element T_j for which

$$T_{j-1} \leq R \leq T_j. \tag{3.5}$$

This way the individuals with a larger fitness will have a higher possibility of being chosen for breeding. Such a method is known as the roulette wheel method.

However, there are disadvantages to using fitness in this way. Using the normal selection method can lead to a few individuals with good fitness early on. These tend to dominate the population, leading to premature convergence, possibly to an incorrect solution. Also in later stages we might find a population where many individuals with the average fitness are close to those with best fitness, leading to more average individuals being able to survive. This is also unwanted and so a re-scaling of fitness is required.

Each individual i receives a rank r_i based on the individual's fitness S_i, with the best fitness having rank 1. Then a new fitness is devised as

$$S_i' = \frac{np - r_i + 1}{np}. \tag{3.6}$$

Thus a relative fitness is used instead of an actual fitness value, leading to a more varied and useful population, at the expense of convergence.

Now that parents have been chosen from the population, the next step is the crossover operation. Let us take two parents with chromosomes

$$\text{chromosome}_1 = c_{11}\, c_{12}\, \cdots\, c_{1n} \tag{3.7}$$

$$\text{chromosome}_2 = c_{21}\, c_{22}\, \cdots\, c_{2n}. \tag{3.8}$$

Then we choose a random number $k \in (1, n)$ and cut the chromosomes between elements $k-1$ and k

$$\text{chromosome}_1 = c_{11}\, c_{12}\, \cdots\, c_{1k-1}|c_{1k}\, \cdots\, c_{1n} \tag{3.9}$$

$$\text{chromosome}_2 = c_{21}\, c_{22}\, \cdots\, c_{2k-1}|c_{2k}\, \cdots\, c_{2n} \tag{3.10}$$

then exchanging the two ends after the cutting point results in two offspring

$$\text{chromosome}_1 = c_{11}\, c_{12}\, \cdots\, c_{1k-1}\, c_{2k}\, \cdots\, c_{2n} \tag{3.11}$$

$$\text{chromosome}_2 = c_{21}\, c_{22}\, \cdots\, c_{2k-1}\, c_{1k}\, \cdots\, c_{1n}. \tag{3.12}$$

Using a large enough population and a long time span we should end up with parameters which sufficiently fit the data. Unfortunately we do not have the benefit of such a large time frame and so improved methods need to be explored to reduce

the time required. For instance having fewer individuals in the population would speed it up, computationally speaking. However, this can create the equivalent of inbreeding. In our problem variety will plummet and we may end up with many individuals being similar to each other. The genetic algorithm requires diversity to move forward. Without it progress stalls.

In genetics there is a chance that certain genes will mutate. Some mutations will be beneficial to the individual, or to the species, whereas others may be detrimental to the individual and will most likely disappear during the selection process of subsequent iterations. We need to recreate the ability to mutate and so a mutation constant p_{mut} is defined. Each gene (i.e. integer in the chain of numbers making up the chromosome) of the offspring is given the probability p_{mut} to mutate. If it does mutate, the integer is converted to another in the interval $(0, 9)$. The value of p_{mut} will depend on the current situation.

Mutation is a way to keep variety up. When diversity is high a high mutation rate is not required, and vice versa. Hence it seems natural to use variation as means to adjust p_{mut}. The cost functional $\mathcal{J}$ is used as a measure of fitness, so define

$$\Delta\mathcal{J} = \frac{\mathcal{J}(r = np/2) - \mathcal{J}(r = 1)}{\mathcal{J}(r = np/2) + \mathcal{J}(r = 1)} \tag{3.13}$$

to be a measure of variety in the current population. As $\Delta\mathcal{J}$ decreases, we know to increase p_{mut}, and vice versa, thus preserving variability.

Finally there is only one more aspect of the genetic algorithm to look at - known as elitism. There are different methods on how to create a new generation of offspring depending on how the offspring will replace the parents. Some methods involve the two offspring replacing the parents with the lowest fitness, or two random parents, before the next crossover begins. Instead the parents are deleted until a population of offspring of size equal to the parents is achieved, then the offspring replace the parents, ready to become the parents of the next generation. However there is no guarantee the offspring will fit the data better than their predecessors therefore the parent which best fits the data is allowed to survive to the next generation, replacing one offspring of lower fitness in the process.

An argument could be made for the best parent to replace the offspring with the worst fitness, or use the roulette method based on fitness as described during the

crossover stage, only with the individuals of lower fitness being prioritised during the selection process. This could improve convergence but that is generally the aim of the crossover stage. Also it could impact the variability of the population which would interfere with the operation of the mutation step. Elitism mainly exists to prevent radical divergence and so in order to retain some independence for each of the 3 stages the parent replaces a randomly chosen offspring.

Due to the random nature of this method there are similarities to a Bayesian approach, such as those that can be found in [49]. Take the random walk approach as an example. At each stage a new possible solution is posed based on the current location. Then it has an acceptance probability assigned to it - this is used to decide whether it is accepted as the next stage in the walk or rejected. Then the method either continues from it or the initial current location based on acceptance. In the genetic algorithm there is a probability imposed at the crossover stage - a choosing of the proposed parents. So it is similar but the stages are placed in a different order - the probability being used to select fit individuals rather than determining the fitness of a proposed solution. Also [13] uses a Bayesian method to image voids using gravimetry, so low density is sought for instead.

To illustrate this in action a simple set up of only 432 voxels was used. An object made of lead was placed in the cargo container and an effort made to find it using only 10 individuals. Figures 3.1 and 3.2 show the minimum of the cost functional and the maximum over subsequent iterations.

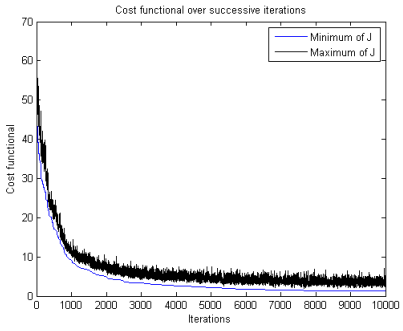

Figure 3.1: Cost functional of the first 10000 iterations for 432 voxels and a population of 10.

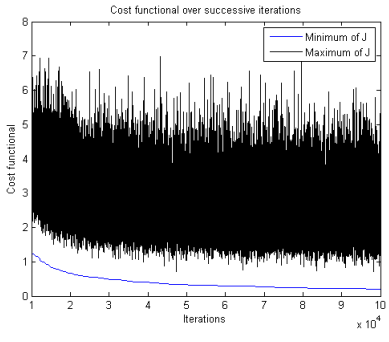

Figure 3.2: Cost functional of iterations 10000 to 100000 for 432 voxels and a population of 10.

Due to elitism the minimum doesn't increase, whereas the maximum can oscillate. Convergence is relatively fast at first, given the random starting point. Unfortunately

it slows down drastically, the minimum cost functional becoming increasingly flat and, in this case, taking 90000 iterations to reduce the cost functional by a factor of 10, as opposed to the first 1000 almost achieving the same reduction.

There are ways to possibly improve convergence. Using a larger population should help matters since a population of 10 is relatively small, but time is also an issue. This example took approximately 5 seconds per 1000 iterations. Increasing the population would increase the time required, but as said above this is a relatively simple example with relatively few voxels. Expanding to more accurate examples would take too long. The advantage of genetic algorithms is their ability to explore other possible solutions, but at the cost of convergence speed. Therefore speed will have to be gained elsewhere if this method is to be useful.

3.1.1 Reparameterisation of the problem

One route to gaining speed is to reduce the number of unknowns. In a pixel-based inversion technique this could be simply reducing the number of voxels, but that would also reduce the resolution. Instead of having the number of unknowns being the number of voxels we can re state the problem to create dependence on a different set of unknowns, one which has a much smaller size [61].

In level set terms each level set function could be represented as

$$\phi(\boldsymbol{u}, \boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{u}, \boldsymbol{x}) \tag{3.14}$$

where $\boldsymbol{u}$ represents the unknown parameters and $\boldsymbol{x} \in \Omega$ are the coordinates of the voxels.

Radial basis functions are a possible choice here, which are discussed in [15], [82] and in greater detail in [16]. [3] uses this method, and chooses adaptive radial basis functions for use in the level set method. The type it uses are known as Wendland's functions. Here we use functions of a Gaussian type instead. Both [64] [20] go into detail about other radial basis functions such as polynomials and multiquadratics and [39] looks into the use of RBFs in a global optimisation scheme, which the genetic algorithm is one of.

The functions we use are of the form

$$f(a, \boldsymbol{b}, c, \boldsymbol{x}) = a \exp\left(-\frac{1}{2c^2}\|\boldsymbol{x} - \boldsymbol{b}\|_2^2\right) \tag{3.15}$$

where $\boldsymbol{u} = (a, \boldsymbol{b}, c)$ are the unknowns. $a, c \in \mathbb{R}$ and $\boldsymbol{b} \in \mathbb{R}^3$ resulting in 5 unknowns for each RBF. Some alterations could be made to the above form in order to change this. c could be linked to a in some way to form a function closer in structure to the normal distribution to reduce the unknowns, or we could state that $\boldsymbol{c} \in \mathbb{R}^3$ to allow a value for every coordinate direction. Also $\boldsymbol{b}$ could remain fixed instead.

In its above form the unknown $\boldsymbol{b}$ is the location of the peak of the function, a is it's height and c controls it's range. Since such functions are always positive, a will have the ability to become negative, thereby providing us with the required domain. For instance if two functions overlap, one has a positive a the other a negative a, the magnitude of each respective a will decide where the boundary of the domain will lie.

Using only a few Gaussian functions will lead to a reconstruction with more spherical objects. Increasing the number of Gaussian functions will allow for more complex shapes, therefore choices will have to be made regarding resolution of images. Combined with the fact we require several level set functions to fill the population of the genetic algorithm, we have several functions to keep track of.

For now we will restrict to simpler cases in which only 2 domains are present and so only 1 level set function is required for each individual. 10 test examples were chosen and the results have been shown in tables 3.1 to 3.8. All were performed on a set up where each voxel was 0.1m length, the sensors were 1m apart and set 0.1m from the outside of the cargo container. A single lead object of size 0.2m was placed at a random location for each case, surrounded by a constant density of 1g/cm^3 and the cost functional was measured after 100 iterations and 1000 iterations. After completing 1000 iterations, the average time was stored for each set up. Then in order to provide some qualitative results the presence of the object and whether it was found in the correct place have been recorded in the final two columns.

Looking at the tables we can come to some conclusions. As expected using more Gaussian functions seems to improve detection as shown in tables 3.1 to 3.4. Using 5 Gaussian functions and 10 individuals seems to have a high success rate. Looking at the tables using 50 individuals (tables 3.5 to 3.8) we see that the object has been detected in all cases and there were only 4 cases where the location of the object wasn't found to within an allowable tolerance. However these have been labelled with a * because an object was recovered in the correct location, except that a second object

was also found in a separate location.

Using more Gaussian functions can increase resolution but it seems they can also recover false objects. Those cases with false objects still had relatively low cost functionals, and so the presence of the second object doesn't impact the fitness in a meaningful way. If we are expecting smaller and fewer objects of a certain type then the use of fewer Gaussian functions is preferable as a form of regularisation.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	2.4721×10^{-4}	1.7916×10^{-4}	Yes	Yes
2	4.2021×10^{-5}	3.6469×10^{-5}	Yes	Yes
3	8.4372×10^{-5}	1.9996×10^{-5}	Yes	Yes
4	6.6×10^{-3}	2.2×10^{-3}	Yes	Yes
5	5.5748×10^{-4}	1.3968×10^{-4}	Yes	Yes
6	5.7244×10^{-4}	7.9877×10^{-5}	Yes	Yes
7	2.7158×10^{-4}	1.888×10^{-5}	Yes	Yes
8	3.329×10^{-4}	3.329×10^{-4}	No	No
9	2.4371×10^{-4}	2.4521×10^{-5}	Yes	Yes
10	6.6735×10^{-5}	1.8689×10^{-5}	Yes	Yes

Table 3.1: Results using 2 Gaussian functions and 10 individuals. Average time taken= 134 seconds.

Loch.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	2.4721×10^{-4}	2.4721×10^{-4}	No	No
2	2.113×10^{-4}	2.6855×10^{-5}	Yes	Yes
3	8.4372×10^{-5}	1.8508×10^{-5}	Yes	Yes
4	6.6×10^{-3}	8.0389×10^{-4}	Yes	Yes
5	7.2522×10^{-4}	1.2262×10^{-4}	Yes	Yes
6	5.7244×10^{-4}	5.7178×10^{-4}	Yes	No
7	2.7158×10^{-4}	1.6628×10^{-5}	Yes	Yes
8	3.329×10^{-4}	1.9428×10^{-5}	Yes	Yes
9	2.4371×10^{-4}	5.6060×10^{-5}	Yes	Yes
10	6.6735×10^{-5}	4.816×10^{-5}	Yes	No

Table 3.2: Results using 3 Gaussian functions and 10 individuals. Average time taken= 158 seconds.

The tables also highlight a disadvantage of the genetic method. Many of the examples have a low cost functional at 100 iterations, some of which are the same at 1000 iterations. Example number 4 in table 3.3 has not been detected in this scenario even after 1000 iterations. In fact, using the cost functional value of 6.6×10^{-3} as a

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	2.4721×10^{-4}	1.6562×10^{-4}	Yes	Yes
2	2.1904×10^{-5}	1.8612×10^{-5}	Yes	Yes
3	2.6143×10^{-5}	1.9061×10^{-5}	Yes	Yes
4	6.6×10^{-3}	6.6×10^{-3}	No	No
5	4.8330×10^{-4}	1.2906×10^{-4}	Yes	Yes
6	5.7244×10^{-4}	5.8016×10^{-5}	Yes	Yes
7	2.7158×10^{-4}	2.7101×10^{-4}	Yes	No
8	3.3252×10^{-4}	7.5118×10^{-5}	Yes	Yes
9	1.3445×10^{-4}	2.0399×10^{-5}	Yes	Yes
10	3.2498×10^{-5}	1.5422×10^{-5}	Yes	Yes

Table 3.3: Results using 4 Gaussian functions and 10 individuals. Average time taken= 185 seconds.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	1.7258×10^{-4}	1.9339×10^{-5}	Yes	Yes
2	2.113×10^{-4}	3.9943×10^{-5}	Yes	Yes
3	5.4023×10^{-5}	1.8263×10^{-5}	Yes	Yes
4	6.6×10^{-3}	1.768×10^{-4}	Yes	Yes
5	7.2522×10^{-4}	9.9052×10^{-5}	Yes	Yes
6	4.2745×10^{-4}	6.1631×10^{-5}	Yes	Yes
7	2.7158×10^{-4}	6.6713×10^{-5}	Yes	Yes
8	3.329×10^{-4}	3.4019×10^{-5}	Yes	Yes
9	1.3873×10^{-4}	7.154×10^{-5}	Yes	Yes
10	6.6735×10^{-5}	2.2936×10^{-5}	Yes	Yes

Table 3.4: Results using 5 Gaussian functions and 10 individuals. Average time taken= 203 seconds.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	2.4721×10^{-4}	2.3646×10^{-5}	Yes	Yes
2	2.113×10^{-4}	1.8361×10^{-5}	Yes	Yes
3	5.7933×10^{-5}	1.8263×10^{-5}	Yes	Yes
4	6.6×10^{-3}	4.417×10^{-5}	Yes	Yes
5	7.2522×10^{-4}	2.1046×10^{-5}	Yes	Yes
6	1.0668×10^{-4}	2.3122×10^{-5}	Yes	Yes
7	1.0024×10^{-4}	1.888×10^{-5}	Yes	Yes
8	3.329×10^{-4}	1.9428×10^{-5}	Yes	Yes
9	2.1614×10^{-4}	1.829×10^{-5}	Yes	Yes
10	6.6735×10^{-5}	2.1359×10^{-5}	Yes	Yes

Table 3.5: Results using 2 Gaussian functions and 50 individuals. Average time taken= 641 seconds.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	1.0795×10^{-4}	3.2931×10^{-5}	Yes	Yes
2	2.166×10^{-5}	1.81×10^{-5}	Yes	Yes
3	2.008×10^{-5}	1.825×10^{-5}	Yes	Yes
4	2.7×10^{-3}	4.754×10^{-5}	Yes	Yes
5	7.2387×10^{-4}	3.8541×10^{-5}	Yes	Yes
6	9.9621×10^{-5}	1.9419×10^{-5}	Yes	Yes
7	2.7086×10^{-4}	1.9165×10^{-5}	Yes	Yes
8	2.6742×10^{-4}	1.9614×10^{-5}	Yes	Yes
9	7.3377×10^{-5}	1.8253×10^{-5}	Yes	Yes
10	4.9247×10^{-5}	1.8325×10^{-5}	Yes	Yes

Table 3.6: Results using 3 Gaussian functions and 50 individuals. Average time taken= 748 seconds.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	5.1007×10^{-5}	3.4117×10^{-5}	Yes	Yes
2	5.2465×10^{-5}	1.5974×10^{-5}	Yes	Yes
3	8.4372×10^{-5}	1.8381×10^{-5}	Yes	Yes
4	6.6×10^{-3}	5.9271×10^{-5}	Yes	Yes
5	1.3303×10^{-4}	5.6378×10^{-5}	Yes	Yes
6	2.1436×10^{-4}	4.7199×10^{-5}	Yes	Yes
7	1.2916×10^{-4}	3.0319×10^{-5}	Yes	Yes
8	3.329×10^{-4}	1.7125×10^{-5}	Yes	Yes
9	2.0399×10^{-5}	1.9199×10^{-5}	Yes	Yes
10	2.6956×10^{-5}	1.7206×10^{-5}	Yes	No*

Table 3.7: Results using 4 Gaussian functions and 50 individuals. Average time taken= 851 seconds.

Example.	$\mathcal{J}$ at 100 iterations	$\mathcal{J}$ at 1000 iterations	Object found?	Location found?
1	4.4781×10^{-5}	1.6364×10^{-5}	Yes	Yes
2	3.1004×10^{-5}	1.8933×10^{-5}	Yes	Yes
3	7.4269×10^{-5}	1.8261×10^{-5}	Yes	Yes
4	6.6×10^{-3}	9.4412×10^{-4}	Yes	Yes
5	7.2522×10^{-4}	3.2075×10^{-5}	Yes	Yes
6	9.1197×10^{-5}	1.6867×10^{-5}	Yes	Yes
7	1.3063×10^{-4}	1.6641×10^{-5}	Yes	No*
8	1.3476×10^{-4}	2.8204×10^{-5}	Yes	No*
9	2.4371×10^{-4}	1.869×10^{-5}	Yes	No*
10	3.0086×10^{-5}	1.7426×10^{-5}	Yes	Yes

Table 3.8: Results using 5 Gaussian functions and 50 individuals. Average time taken= 963 seconds.

guide, it has not been detected at 100 iterations for many other set ups. The reason for this is it's location. Figure 3.3 shows that it is 0.6m from the edge of the cargo container in the y-direction, which is half way between two sensor gates and so has a reduced effect on the sensor.

Similarly location number 1 is at the same distance in the y-direction (see figure 3.4). After 100 iterations the cost function is $\mathcal{J} = 2.4721 \times 10^{-1}$ and in fact the object hasn't been found at that stage. After 1000 iterations it has been found, and the reconstruction is shown in figure 3.5. In contrast to this, look at table 3.2. There the object at location 4 has been found and the one at location 1 has not. Additionally, due to the slight random nature of the algorithm, if we were to run it again we could get different results, at least at certain stages of the method.

In the previous level set method in which the steepest descent direction was implemented, if a point was reached where no step length could be found in a certain interval to reduce the cost functional then the method stopped. A checking mechanism such as that is difficult to apply to the genetic method as the evolution of the cost functional becomes remarkably flat as the algorithm is run. In fact, due to elitism, there can be several consecutive iterations where the minimum cost functional doesn't change. A tolerance level for the cost functional may be the best course of action. Once we reach it the method ceases.

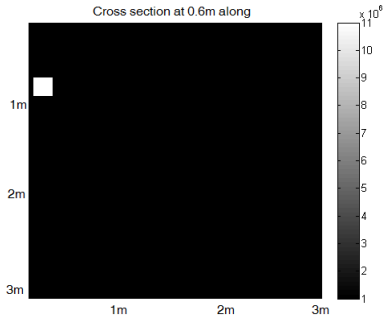

Figure 3.3: One slide of the actual location of example 4 from the tables.

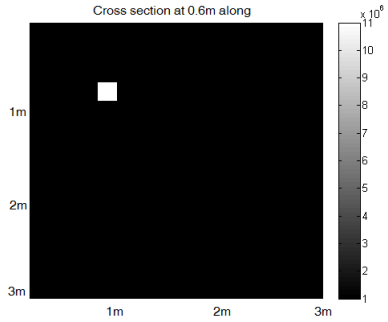

Figure 3.4: One slide of the actual location of example 1 from the tables.

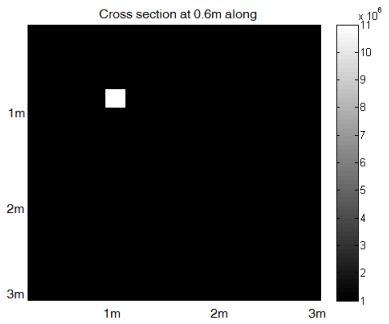

Figure 3.5: One slide of the reconstruction location of example 1 from the tables using 10 individuals and 2 Gaussian functions.

3.1.2 Genetic algorithm and colour level set approach with radial basis functions

Now that we have looked at the simple case of one object, it is time to extend it to a more complex case. For this we return to the use of the colour level set method, which has been described in section 2.6. Each level set function a comprised of the Gaussian functions from the previous section. Using the genetic algorithm it is relatively straightforward to produce some results, such as those in figure 3.6.

Before analysing the results there are a few things to note. First is that even though this is a more cluttered example it is still relatively simple. It only has three objects within it and they don't touch each other. This can influence the choice of parameters as only 5 Gaussian functions were used for each level set function due to the low number of objects to resolve. Using more could lead to more objects appearing, but this is prior information we wouldn't have access to.

Also this result was designed to be a best case scenario. For that reason a population of 50 was used (i.e. there are 50 density distributions at each stage of the genetic algorithm) and it was run for 10000 iterations, which took several hours and so is not practical for a real world scenario.

On the subject of the parameters they have to be constrained in order to give the results we expect. Parameter $a \in (-1, 1)$ to balance positive and negative possibilities as this seems to suffice for our needs. $\boldsymbol{b}$ is simply constrained by the dimensions of the cargo container. If we wanted to focus on one region then we could, or have them fixed to a specific location. c is related to the radius of the sphere created by each Gaussian function. Increasing it leads to larger objects, but here it is in the interval

$(0, 1)$.

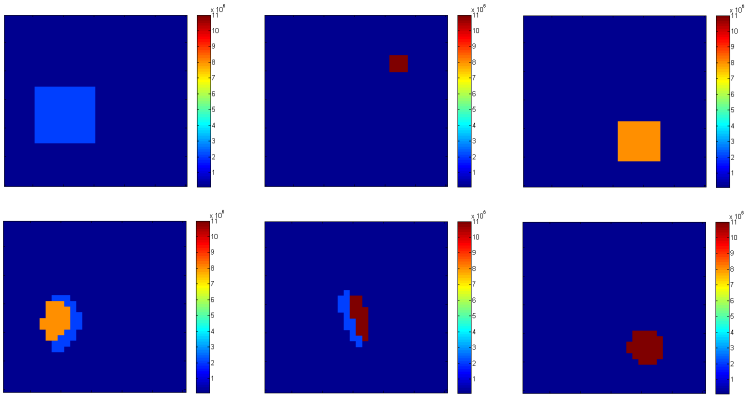

Figure 3.6: Top layer is actual location of objects. Bottom layer is reconstructed location. Results shown are after 10000 iterations. Each image is a cross section of the cargo container measuring 3m $\times$3m.

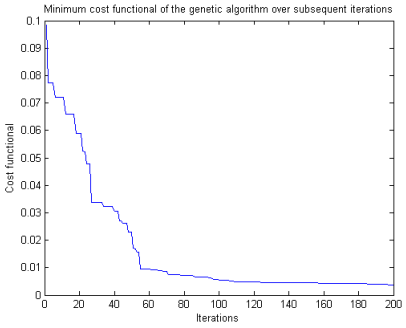

Figure 3.7: Minimum cost functional of the genetic algorithm over the first 200 iterations.

In figure 3.6 the actual density distribution is on the top and the reconstruction is on the bottom, lined up to make it easier to compare the two. The wooden box has been instead reconstructed as a combination of wood and metal. This isn't too much of a problem as we don't care about wood or metal. The lead has been reconstructed slightly further away from the actual location, and with some metal too. However getting the location correct isn't as much of a priority. The problem arises when we look at the metal object, which has been reconstructed as lead, and so false positives can arise.

Here the results illustrate one of two possible features - either the non-uniqueness of the solution or the difficulty in determining the convergence of the genetic algorithm. As stated before the size and density of an object are interchangeable as far as gravity

is concerned. The results seen here may give the same fit to the data as others that are closer to a correct solution. In this case the results have lead to a false positive, but it could have just as easily lead to a false negative instead.

This also illustrates a potential strength of the genetic algorithm, as it is theoretically possible for it to find any and all possible solutions to the problem, and due to the random nature a rerun wouldn't necessarily lead to the same result. Perhaps running several times would provide us with more information, which forms the basis of the ensemble algorithm explored in Chapter 5.

Also mentioned is the lack of knowing when convergence occurs. When using the genetic algorithm it becomes remarkably slow in the later stages. It should eventually converge but there is no guarantee on when that might happen - there is not a known length scale by which to measure it. That is the reason for attempting to combine it with other methods - in order to avoid getting stuck near false solutions.

The steepest descent algorithm works directly on the previous case and so it is easy to track the evolution of the reconstruction from iteration to iteration. Usually with the genetic algorithm we look at the reconstruction which currently fits the data the best. So if we are tracking the best reconstruction at each iteration it might suddenly change completely when a better reconstruction is found - this is how it avoids local minima.

Given the random nature of the genetic algorithm there is no guarantee that similar results will arise if ran again. In fact it is highly unlikely that the results will be the same. Both the genetic algorithm and the steepest descent method have their strengths in different areas.

3.2 Sparsity

We have looked at a few norms so far when attempting to minimise the data misfit. An L_2-norm acting on the voxel-based density distribution $\boldsymbol{\rho}$ will provide us with a relatively smooth reconstruction. An L_1-norm will allow for larger jumps in the density leading to larger magnitudes and gradients. In order to find a sparse solution we require the L_0-norm. There are methods we could use, such as the iterative thresholding method as explored in [26]. Also it is possible that a solution is sparse in a certain

dictionary, which is why papers such as [28] and [21] use wavelets, whereas [18] looks into the use of ridgelets and curvelets along with total variation regularisation.. An alternative method is Basis Pursuit which focuses more on the L_1-norm, which is mentioned in [14] and expanded upon in [23]. The use of the L_0-norm allows larger jumps between components, forcing smaller components to become zero and therefore promote sparsity. Another method is gradient projection as explored in [34].

This isn't quite a norm in the same sense as the others as it doesn't minimise the magnitude or gradients of $\boldsymbol{\rho}$ in any way. Other norms are measurements on the values of individual components of ρ. Broadly speaking the L_2-norm smooths out ρ whereas the L_1 norm allows for larger jumps between components, thereby enforcing sparsity. In its simplest form the L_0-norm enforces sparsity via a different avenue - by restricting the number of non-zero elements of ρ. Methods can then be employed to minimise the data misfit as long as this restriction is enforced. It can amount to minimising the data misfit using a steepest descent but with most components known to be zero. Given fewer non-zero elements a larger density can be chosen for that region. So this allows us to find objects of larger density with lower volume, which is our are of interest.

However, the high density material isn't the only object that will be in the cargo container (if there is any at all). There should be other objects in the background. While we aren't looking for any density lower than lead such objects will contribute towards the data - larger objects tend to have an impact even if they have relatively low density. So we need to try and reconstruct them to an appropriate degree without falsely reconstructing them as higher density objects.

First we split the density into two parts

$$\boldsymbol{\rho} = \boldsymbol{\rho}_1 + \boldsymbol{\rho}_2 \tag{3.16}$$

where $\boldsymbol{\rho}_1$ represents a sparse reconstruction and $\boldsymbol{\rho}_2$ is a smooth reconstruction (allowed for the background because we don't need a high resolution for that). The general aim is to find an algorithm which can alternate between the smooth and sparse minimisations to obtain a reconstruction with a high density object (and doesn't find it when there is nothing to be found).

Before going into the methods explored for the sparse reconstruction we first look at how to find the smooth reconstruction $\boldsymbol{\rho}_2$.

3.2.1 Steepest descent combined with sparsity

The purpose of this section is for the steepest descent method to work on the smooth density distribution $\boldsymbol{\rho}_2$, leaving the orthogonal matching pursuit to work on the sparse reconstruction. However other work has been performed on using a gradient technique and the L_1 [47], which involves the use the Barzilai and Borwein step size [7].

As defined previously the cost functional is the minimisation of the L2-norm of the residual

$$\mathcal{J}(\boldsymbol{\rho}) = \frac{1}{2} \langle \mathbf{G}\boldsymbol{\rho} - \mathbf{d}, \mathbf{G}\boldsymbol{\rho} - \mathbf{d} \rangle . \tag{3.17}$$

The method of steepest descent attempts to find an alteration $\boldsymbol{\delta\rho}$ such that

$$\mathcal{J}(\boldsymbol{\rho} + \boldsymbol{\delta\rho}) < \mathcal{J}(\boldsymbol{\rho}). \tag{3.18}$$

Using the same steps as defined previously if we choose

$$\boldsymbol{\delta\rho} = -\mathbf{G}^T(\mathbf{G}\boldsymbol{\rho} - \mathbf{d}) \tag{3.19}$$

then (3.18) should hold. Therefore at each stage of the iteration we have

$$\boldsymbol{\rho}_2^{n+1} = \boldsymbol{\rho}_2^n + \tau \boldsymbol{\delta\rho}^n \tag{3.20}$$

for a suitable step length τ. Note the use of subscript 2 here as we are only adding a change to $\boldsymbol{\rho}_2$.

The overall algorithm will alternate between enacting several iterations of the smoothing method, followed by one iteration of one of the sparse methods described in the next sections.

3.2.2 Iterative shrinkage thresholding algorithm

A simple sparse method is known as iterative shrinkage thresholding which begins by first using the update described in the steepest descent method section, and then altering is slightly. For $\boldsymbol{\rho}_1$ sparse reconstruction, at each iteration enact

$$\boldsymbol{\rho}_1^{n+1} = \mathcal{T}_{\lambda\tau}(\boldsymbol{\rho}_1^n + \tau \boldsymbol{\delta\rho}^n) \tag{3.21}$$

where

$$\mathcal{T}_\alpha(\boldsymbol{\rho}) = \max(0, |\boldsymbol{\rho}| - \alpha)\mathrm{sgn}(\rho_i) \tag{3.22}$$

and figure 3.8 illustrates the outcome of this.

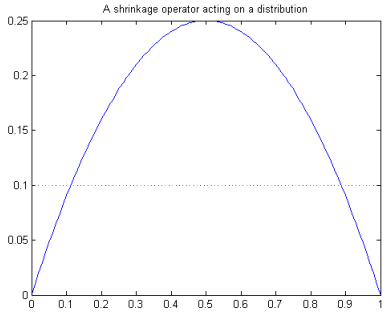

Figure 3.8: The shrinkage operator is defined by the horizontal line. Anything below it is set to zero, and it then becomes the new x-axis.

There are a couple of main types of thresholding, known as soft and hard thresholding. Hard thresholding looks for a specific sparsity level and adjusts α until it finds it i.e. the number of non-zero elements is known beforehand. Soft thresholding instead chooses α at the start of the iterative technique and allows the method to run, thus allowing for different sparsity levels. The prior information we have access to is limited and so soft thresholding has been chosen for this algorithm.

Adaptive thresholding is explored in [12] and [5], which also uses a combination of L_0 and L_2 regularisation. [8] works on increasing the global convergence rate of the iterative thresholding algorithm. Similarly [10] works to increase convergence using a 2-step method.

Thresholding is used in many situations and it should suit our problem relatively well. However there were some shortcomings as there was difficulty in recovering large magnitudes which may be a failing of a particular stage of the algorithm. First the steepest descent direction $\delta\phi$ is found. Then a Landweber iteration is enacted, which includes a line search to find the optimal τ. Finally the shrinkage operator acts, using $\lambda\tau$ as the threshold.

While this last part allows for sparse solutions it also reduces the magnitude of the solution, to the point at which the sparse solution did not stand out enough from the smooth solution $\boldsymbol{\rho}_2$, making the overall density distribution $\boldsymbol{\rho} = \boldsymbol{\rho}_1 + \boldsymbol{\rho}_2$ also smooth. Our ability to combat this is lessened somewhat due to the shrinkage operator being $\alpha = \lambda\tau$. λ is defined beforehand but τ may change magnitude from one iteration to

the next. In the end a more complex method was sought after.

3.2.3 Orthogonal matching pursuit

Another area to look at is a group of algorithms called 'greedy algorithms'. Why they are called that should become apparent when looking at the algorithm. There has been the Matching Pursuit algorithm developed [60] but we focused on one known as Orthogonal Matching Pursuit [69], which works only on the sparse part of the distribution $\boldsymbol{\rho}_1$ of the density distribution, alternating between this and the optimisation of the smooth solution. A more in depth analysis of this and sparsity in general is explored in [4]. For the purposes of simplicity for the time being let us assume that we are dealing with a single density distribution $\boldsymbol{\rho}$, which will be a sparse reconstruction with a given sparsity level that optimises

$$J = \frac{1}{2}\|\mathbf{G}\boldsymbol{\rho} - \mathbf{d}\|_2^2 \tag{3.23}$$

as best it can.

Denote $\boldsymbol{\rho}^k$ to be the density distribution at sparsity level k. Then the overall method is explained in algorithm 4 and expanded upon after.

Algorithm 4 Orthogonal Matching Pursuit

procedure FIND $\boldsymbol{\rho}$ THAT MINIMISES $\frac{1}{2}\|G\boldsymbol{\rho} - \boldsymbol{d}\|_2^2$ SUBJECT TO $\|\boldsymbol{\rho}\|_0 = n$
  $k = 0$.
  Set initial solution $\boldsymbol{\rho}^0 = 0$.
  Set initial residual $\mathcal{R}^0 = \boldsymbol{G\rho^0} - \boldsymbol{d}$.
  Set initial support $\mathcal{S}^0 = Support(\boldsymbol{\rho^0}) = \emptyset$.
  repeat
    $k = k + 1$.
    Compute the errors $\epsilon(j) = min_{z_j}\frac{1}{2}\|\boldsymbol{g}_j z_j + \mathcal{R}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = -\boldsymbol{g}_j^T\mathcal{R}^{k-1}/\|\boldsymbol{g}_j\|_2^2$.
    Find a minimiser j_0 of $\epsilon(j)$ such that $\epsilon(j_0) \leq \epsilon(j)$ and j_0 is not currently in $\mathcal{S}^{k-1}$. Then define $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
    Compute $\boldsymbol{\rho}^k$ that minimises $\frac{1}{2}\|G\boldsymbol{\rho}^k - \boldsymbol{d}\|_2^2$ subject to $Support(\boldsymbol{\rho}^k) = \mathcal{S}^k$.
    Update the residual $\mathcal{R}^k = G\boldsymbol{\rho}^k - \boldsymbol{d}$.
  until n iterations completed

Initialise the problem by setting $\boldsymbol{\rho}^0 = \mathbf{0}$, so that the current sparsity level is zero. Then the residual is calculated which, at this stage, is simply $\mathcal{R}^0 = \boldsymbol{G\rho}^0 - \boldsymbol{d} = -\boldsymbol{d}$.

Next the sparsity level is increased to 1 and so now we are aiming to find an object of size equal to one voxel. The location and density of said voxel are the current

unknowns to be optimised. Each voxel is assigned a number $j = 1, \ldots, n$ where n is the total number of voxels in the domain Ω. Let us begin by assuming the object is at voxel number 1, then the updated residual is

$$\begin{aligned} \mathcal{R}^1 &= \boldsymbol{G}\boldsymbol{\rho}^1 - \boldsymbol{d} \\ &= \boldsymbol{g}_1 z_1 + \mathcal{R}^0, \end{aligned} \tag{3.24}$$

where z_1 is the density of voxel 1 and $\boldsymbol{g}_1$ is the first column of $\boldsymbol{G}$.

Therefore we need to find the error

$$\epsilon(1) = min_{z_1} \frac{1}{2} \|\boldsymbol{g}_1 z_1 + \mathcal{R}^0\|_2^2 \tag{3.25}$$

which is found at the critical point

$$z_1^* = \frac{-\boldsymbol{g}_1^T \mathcal{R}^0}{\|\boldsymbol{g}_1\|_2^2}. \tag{3.26}$$

Continuing with all voxels in such a way gives us z_j^* for all $j = 1, \ldots, n$ and the corresponding $\epsilon(j)$. Next find the minimum error and its corresponding voxel number i.e. find j^1 such that $\epsilon(j^1) \leq \epsilon(j)$ for all j, and update the support

$$\mathcal{S}^1 = \{j^1\}. \tag{3.27}$$

With the addition of this first non-zero voxel for $\boldsymbol{\rho}$ we update the residual

$$\mathcal{R}^1 = \epsilon(j^1), \tag{3.28}$$

before moving onto the next iteration.

The general step is slightly more complicated than the first one. At stage $k-1$ we have a residual $\mathcal{R}^{k-1} = \boldsymbol{G}\boldsymbol{\rho}^{k-1} - \boldsymbol{d}$ based on a density distribution $\boldsymbol{\rho}^{k-1}$ having a sparsity level of $k-1$, and the voxel numbers having been stored in the support

$$\mathcal{S}^{k-1} = \{j^1, j^2, \ldots, j^{k-1}\}. \tag{3.29}$$

Next we assume the actual sparsity level is k, so we need to find one more voxel to add to the support, by cycling through the remaining voxels not currently in the support. So $\forall j \notin \mathcal{S}^{k-1}$ we calculate

$$\epsilon(j) = min_{z_j} \frac{1}{2} \|\boldsymbol{g}_j z_j + \mathcal{R}^{k-1}\|_2^2 \tag{3.30}$$

using

$$z_j^* = \frac{-\boldsymbol{g}_j^T \mathcal{R}^{k-1}}{\|\boldsymbol{g}_j\|_2^2}, \tag{3.31}$$

then find j^k such that $\epsilon(j^k) \leq \epsilon(j)\ \forall j \notin \mathcal{S}^{k-1}$. Add this to the support

$$\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j^k\}. \tag{3.32}$$

Up until this point the steps have remained the same as the first iteration. However now we minimise

$$\mathcal{J} = \frac{1}{2}\|\mathbf{G}\boldsymbol{\rho}^k - \mathbf{d}\|_2^2 \tag{3.33}$$

subject to $\sup(\boldsymbol{\rho}^k) = \mathcal{S}^k$. So all non-zero voxels of $\boldsymbol{\rho}$ are flexible, not just the newly added j^k. Therefore the density distribution $\boldsymbol{\rho}^k$ could be very different to that of $\boldsymbol{\rho}^{k-1}$ After performing this step the residual $\mathcal{R}^k = \boldsymbol{G}\boldsymbol{\rho}^k - \boldsymbol{d}$ is stored before moving onto the next step.

This general step illustrates the reason why such algorithms are known as "greedy". When searching for an appropriate component of $\boldsymbol{\rho}$ to add to the support, all voxels within the domain have to be sampled (those not currently part of the support). It may not be much of a problem for the cases shown here but for higher dimensional grids it can become time-intensive. Under such circumstances some approximations may have to be employed. Instead of sampling the whole domain, it could be divided into sub-domains based on the outcome of other methods - a splitting up the method into two stages. The use of a smooth reconstruction can help us identify potential regions to look at. The sparse approach could then be applied to these regions to see if any high density material is present. This was one of the motivating factors behind the analysis performed in the next section.

Returning to our actual density distribution of $\boldsymbol{\rho} = \boldsymbol{\rho}_1 + \boldsymbol{\rho}_2$ the above method only applies to $\boldsymbol{\rho}_1$, the sparse reconstruction. Instead of continually increasing the sparsity of $\boldsymbol{\rho}_1$ the method alternates between finding a sparse solution followed by a smooth solution - a sparse solution of $\boldsymbol{\rho}_1^{k-1}$ is sought while keeping $\boldsymbol{\rho}_2$ fixed before moving onto the smooth solution. After a smooth solution for $\boldsymbol{\rho}_2$ is found while keeping $\boldsymbol{\rho}_1$ fixed (using a gradient descent and line search method along with a smoothing term γ) we go back and look for a sparse solution $\boldsymbol{\rho}_1^k$ of sparsity level k. The method continues until both approaches appear to converge, which is when both the line search method

is unable to find a non-zero step length to alter the smooth solution and no addition to the sparsity level will improve the fitness.

A specific density distribution has been chosen in figure 3.9. When this example was made the location of the fissile material was chosen at random with size 10cm on each side and a density of $18 \times 10^6 \text{g/m}^3$. Then a lead box was placed around it with thickness 20cm and density $11 \times 10^6 \text{g/m}^3$. It sits in slides 45 to 49.

The background is difficult to make out and so it can be seen on its own in figure 3.10. It is periodic, similar to three piles of materials in certain cross sections, repeating every metre or so. The aim was to try and model a realistic background, or at least one that isn't a constant density. The maximum density of the background is $3 \times 10^6 \text{g/m}^3$.

After creating the data from this distribution the method was run. At each iteration the algorithm runs the sparse approach on $\boldsymbol{\rho}$, followed by a steepest descent algorithm incorporating Landweber iterations. The sparsity level increases by one for every ten Landweber iterations completed using the parameter value $\gamma = 1$. Some slides have been chosen below in figures 3.11 and 3.12.

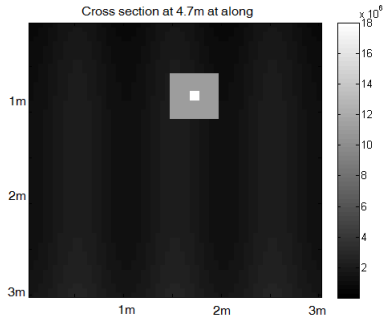

Figure 3.9: Actual density distribution in the region containing the lead box and fissile material.

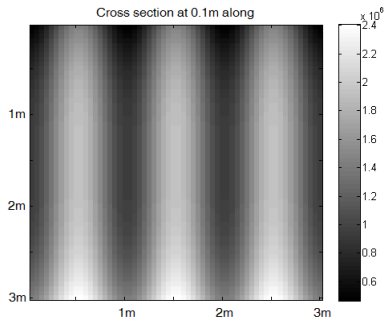

Figure 3.10: Actual density distribution of the background.

The method has had some success. A relatively high density material has been reconstructed in slice 45, which is relatively close to the actual location. It is also a lot smaller than the actual object, but that doesn't matter so much. However figure 3.11 shows the slice with the maximum density in it, which is very far away from the actual.

For some reason instead of creating a clustering of points in certain areas there are individual voxels of high density popping up throughout the cargo container. There is a tendency towards the lower part of the cargo container, which is interesting given

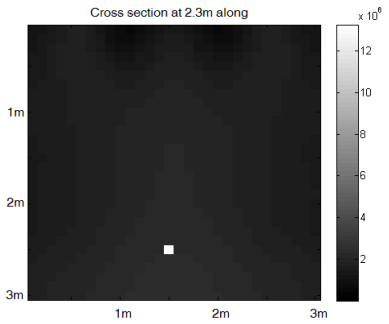

Figure 3.11: Slice with the maximum density recovered with $\gamma = 1$.

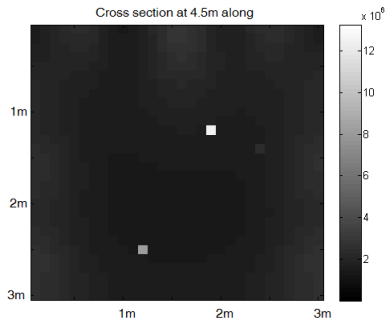

Figure 3.12: Slice with density recovered closest to actual location with $\gamma = 1$.

that there are no sensors on the bottom. So it could be a case of sensor dependency - certain areas are favoured over others regardless of there being high density material there. So perhaps we would need to dampen these areas by enforcing some sort of clustering to occur - which is in broad terms what the level set approach is good at.

Similar results occur under different conditions (for example after varying the parameter γ and for different locations of fissile material) so it is more of a general failing of the method rather than for this specific example. It seems that when the whole domain is available to search for high density material that is what is found.

This was one approach which is not currently performing to standard. There could be alternatives employed such as a model objective function incorporating both respective terms, much like in a Tikhonov regularisation approach studied in [51]. It could be difficult to combine the two norms in the same way but there should be a way to do it. However this is left as an alternative avenue of thought, with more analysis being performed on the current state of the method.

3.2.4 Choosing the potential region for fissile material

Orthogonal matching pursuit fails in certain respects - we need to restrict the region in which we search for fissile material. So we want to say where we expect to find fissile material, with at least some degree of confidence.

For this section the same actual density was used as seen in the previous section. Refer to figure 3.13. This was made by simply running 100 iterations of the Landweber method and not looking for any sparse solutions, which is why the maximum density is much lower than fissile material. However, even though this example is looking for

a smooth solution, the presence of fissile material should still make a difference, and it appears it has.

The largest density occurs at 4.6m along, which is within the intervals of slices in which the fissile material exists. It has been smoothed out and dragged to the edge to be closer to the sensors though. Even though there is a weighting function attempting to compensate for the sensors (based on the distance from the sensors) it doesn't completely do its job. However the maximum density region should be indicative of a high density material, so we should be able to use it as a starting point.

The way in which this was accomplished was as follows. We start by choosing only the voxel with the highest density and use this as our feasible domain (instead of looking at the whole cargo container). Then we place an object of density equal to lead there (so we are restricting the feasible density too) and see how that affects the cost functional. If the cost functional is reduced then we keep that voxel as lead.

If the cost functional has not been reduced by that choice then we expand the feasible domain. Now it is a cube centered on the voxel with the maximum density at the previous point with a thickness in all directions of one voxel (so $3 \times 3 \times 3$ voxels). Repeat the above steps and keep increasing the size of the feasible domain until a useable voxel is found or we reach a prescribed maximum size.

Next we run 10 iterations of the Landweber method, but keep any chosen voxels unchanged (so the landweber method doesn't affect any voxels with high density). Then repeat the above steps again until both methods converge or it reaches the end of its run. The results are shown below in figures 3.14 to 3.17.

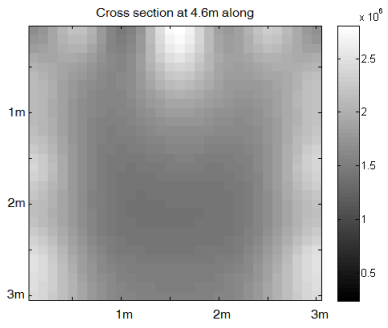

Figure 3.13: Density distribution after 100 iterations of Landweber and no sparse solution looked for.

Figures 3.14 and 3.15 show the cross sections where the method gets closest to the

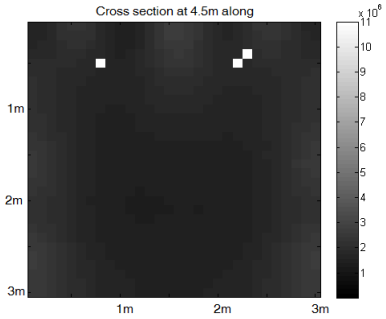

Figure 3.14: One slide close to the actual location of the fissile material.

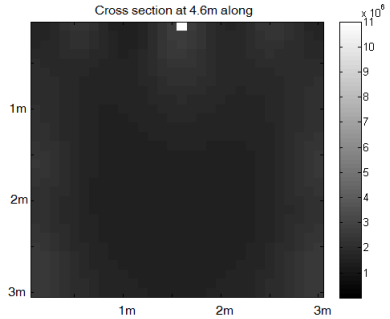

Figure 3.15: Another slide close to the actual location of the fissile material.

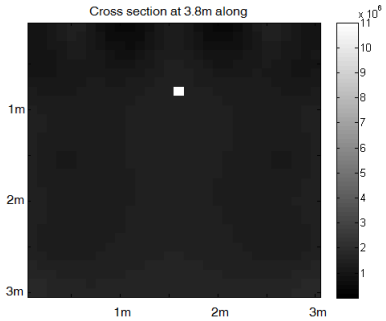

Figure 3.16: Reconstructed density furthest away in the negative y-direction.

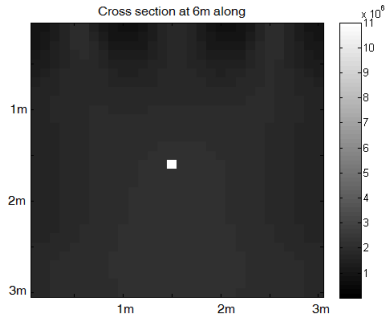

Figure 3.17: Reconstructed density furthest away in the positive y-direction.

actual location. It could be thought of as a good result, but also figures 3.16 and 3.17 show how far away the density has been reconstructed too. An argument could be made that the slightly more clustering in the cross section at 4.5m shows the presence of fissile material, but we can't really be sure. It is an improvement over the previous section's results, but it does have a failing.

Look at figure 3.18. This is the same first stage of the method (100 Landweber iterations) when applied to an example where there isn't any fissile material. The maximum density recovered exists at the bottom edge of the cargo container, a region furthest away from the sensors, which could be due to the weighting function applied at each iteration.

Gravity gradiometry decreases in strength relative to one over distance cubed. Without any weighting all density would be reconstructed near the sensors. In order to counteract this we used a weighting function based on the sensitivity matrix G. By adding up the columns of G we end up with a 'sensitivity' for each voxel. Using the reciprocal of this we obtain a weighting function to apply. It's possible that a

weighting function in this form shouldn't work. It's also possible there is some sort of calculation error creeping in as in this case it seems that the areas furthest away are being prioritised.

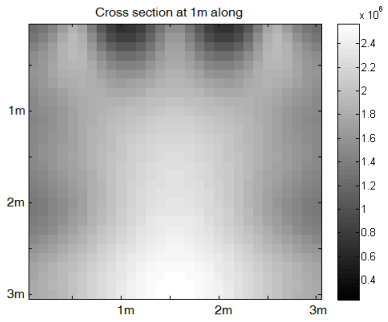

Figure 3.18: Density distribution after 100 iterations of Landweber, no sparse solution looked for and no high density material in the actual distribution.

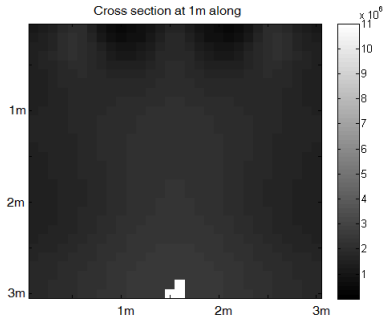

Figure 3.19: Cross section at 1m with high density material falsely reconstructed.

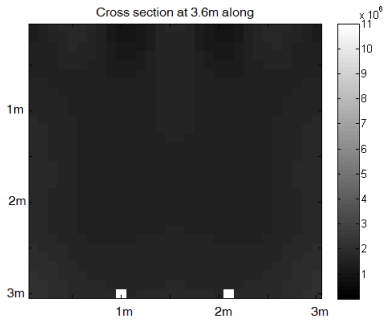

Figure 3.20: Cross section at 3.6m with high density material falsely reconstructed.

The general hope is that the sparse method should avoid any false positives after the Landweber method identifies a region but in figures 3.19 to 3.22 we find this is not the case. High density material has been reconstructed in several places throughout the cargo container. In fact the objects are recovered in similar locations if we run 100 Landweber iterations and then allow sparsity anywhere in the domain, suggesting that such regions attract high density material naturally.

There isn't a reason why the sparse method should be attracted to those regions, but it is only when the Landweber iterative method has been run beforehand. This is usually the difference between gradient-based and sampling-based techniques, as seen when we compared the steepest descent method to the genetic algorithm.

Gradient-based techniques tend to be faster in converging but are more likely to

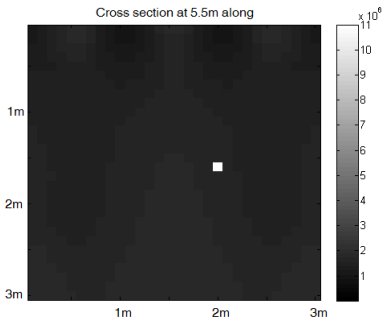

Figure 3.21: Cross section at 5.5m with high density material falsely reconstructed.

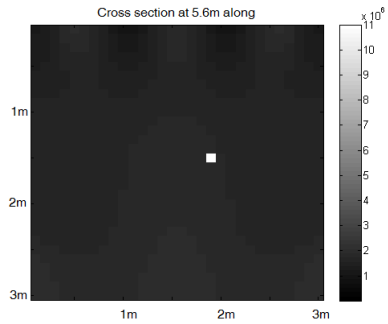

Figure 3.22: Cross section at 5.6m with high density material falsely reconstructed.

get stuck near local minima. Sampling-based techniques can be slow as they usually explore many possibilities in the domain, but they are better adapted to avoiding local minima. Therefore an argument could be made for replacing the gradient-based technique with another sampling-based technique. Then the orthogonal matching pursuit method might have a better staring point.

Also the alterations made to the OMP method are quite restrictive - once it has found a voxel with density equal to lead it will always have that value no matter what. Perhaps alternating between two levels is possible - it can have either lead or a value of zero. A final suggestion could be to add a separate stage where voxels in the sparse solution can be removed too.

3.2.5 Genetic algorithm and OMP

An alternative ties in with work performed before - namely the genetic algorithm. Previously we attempted to use radial basis functions along with a combination of the genetic algorithm and the colour level set method, but it either wasn't going to converge or it was taking too long to converge. It usually had problems identifying the size and density of each object - larger objects of lower density were often reconstructed as smaller objects of higher density, but it is possible this could be avoided by reducing the solution space.

If the genetic algorithm was only looking for a smooth solution then it's possible some failings could disappear. It would have a lower maximum density so lower density objects will be reconstructed closer to what they should be, and higher density objects

would be simply reconstructed as lower density objects. Then the OMP could work on the highest density regions.

The reparameterisation of the problem is another question. Previously level set functions were used because they are good at defining sharp boundaries. If we are looking for a smoother solution then perhaps this is not required. Instead we could just use radial basis functions directly for the background.

If we are to combine two different methods together we need to make sure they are both handled properly. Previously when using the method of steepest descent there was a set number of iterations used in between checking for an addition to the sparsity level. We need to do something similar here, only using more iterations for the genetic method as it takes longer to have any effect.

The general method is as follows. First start by running 1000 iterations of the genetic algorithm to find a good approximation for the smooth solution (i.e. the background), and also a good starting point in which to search for a sparse solution. Then a sparsity level of 1 is assumed and OMP is used. Following on from this another 100 iterations of the genetic algorithm is enacted before moving up to the next sparsity level. A maximum proposed sparsity level of 100 (with 100 iterations of the genetic algorithm in between each sparsity level) was assumed before stopping.

One set of results is shown below. In them the algorithm was run for the actual distribution shown in figures 3.9 and 3.10. 10 radial basis functions were used and the genetic algorithm ran on a population of 20 individuals.

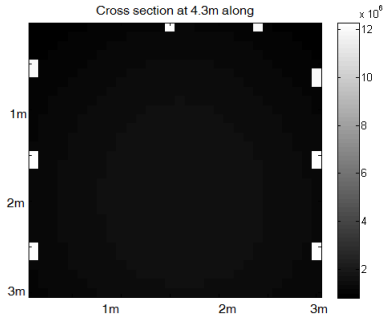

Figure 3.23: Cross section at slice 4.3m of the reconstruction using the genetic algorithm spliced with OMP.

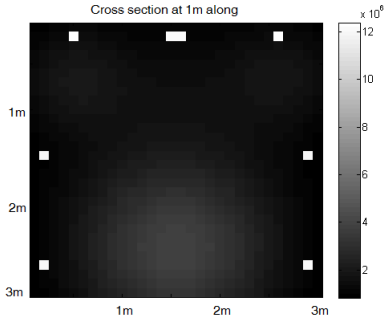

Figure 3.24: Cross section at 1m of the reconstruction using the genetic algorithm spliced with OMP.

The cross section at 4.3m along the cargo container has been included as this is the closest slice of the cargo container to the actual fissile material which has a high

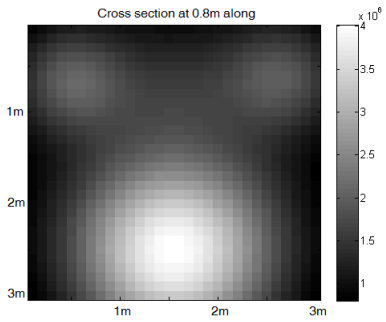

Figure 3.25: Cross section at 0.8 of the background of the reconstruction using the genetic algorithm spliced with OMP.

density in its reconstruction. There are several voxels of high density in this slice, but all are reconstructed close to the sensor locations. On its own this could just be that the fissile material nearby has been pulled towards the sensors, which wouldn't necessarily be a bad result if it reliably told us about its presence. However figure 3.24 shows the cross section at 1m along the cargo container which also has high density material reconstructed near to the sensors. There are other slices like this throughout the cargo container.

One possibility for this is that the genetic algorithm has not performed as well as we need it to. If it doesn't provide us with a good enough smooth reconstruction at the start then this could negatively affect the performance of the sparse method in subsequent iterations. For instance a large volume of low density material would have an impact on the data, and hopefully the genetic algorithm would provide a smooth reconstruction which creates similar data. If it doesn't then the OMP might work to place a high density object there instead. Similarly, if the genetic algorithm has reconstructed a smooth solution in the vicinity of the fissile material, then this could reduce the driving force behind placing high density material there instead - the smooth solution having matched the data enough.

The second possibility is that a sparse solution will always look like this, no matter what reconstruction the genetic algorithm produces. If this is the case then there would be no point in proceeding with looking for sparse solutions. We can first try to eliminate one possibility. The above radial basis functions each have 5 unknowns controlling magnitude, volume and location, all of which are restricted in some way. The parameter controlling magnitude is restricted to keep us in the realm of smooth

solutions. However since the location is allowed to change it is possible for two or more radial basis functions to lie on top of each other, thus increasing the magnitude. This happened in several instances, leading to part of the case described above - the smooth solution becomes less smooth and so interferes with the sparse solution.

In figures 3.26 and 3.27 the same actual density distribution has been used as before and the same algorithm, but now each radial basis function only contains two unknowns relating to magnitude and volume thus fixing each one to a prescribed location. Since there are fewer unknowns this should increase efficiency. However, now that the radial basis functions don't move around the cargo container we have to make sure we have complete coverage, or as close to it as we can get. Therefore the centers of the radial basis functions are placed in the cargo container at specific intervals. Using the same orientation as the slices, the functions are placed in a 3×3 grid to make a slice of functions, then 6 slices are placed throughout the cargo container at fixed intervals. Therefore we are now working with 54 Gaussians as opposed to 10 before, which means 104 unknowns as opposed to 50 before.

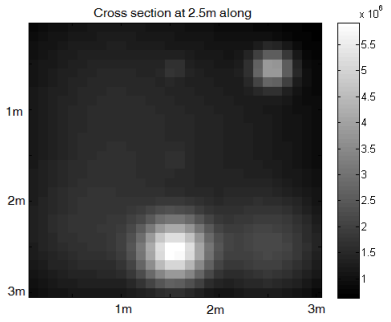

Figure 3.26: Cross section at 2.5m along of the background of the reconstruction using the genetic algorithm spliced with OMP and radial basis functions having fixed positions.

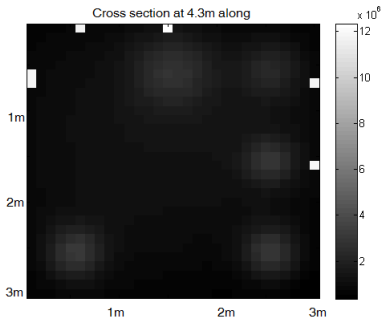

Figure 3.27: Cross section at 4.3m along of the reconstruction using the genetic algorithm spliced with OMP and radial basis functions having fixed positions.

Figure 3.26 shows the background distribution of the reconstruction. It might be easier to view the function setup described above using this as it might be indicative of the 3×3 layout of the functions. Similar slices appear throughout the cargo container, and this slice contains the highest density of the background, which is close to 6×10^6. The background density doesn't fit the actual background density very well, but then it isn't supposed to, only emulate the data good enough to obtain the sparse

reconstruction.

Figure 3.27 shows the slice of the reconstruction closest to the fissile material which contains high density material in the reconstruction. It is the same slice as in figure 3.23, which is unsurprising given the sensors are nearby. However the reconstruction is slightly different as the voxels chosen to be non-zero have more of a clustering around the top right region of the cargo container. This could be an indication that the method could be moving more towards the fissile material. There are slices throughout the cargo container which have similar structures to that seen in figure 3.24, but this might be a step in the right direction.

3.2.6 Finding fissile material in the correct location

From the results so far it appears that some regions of the domain may be favoured over others, for instance the regions near the sensors. It may be advantageous to track where fissile material is likely to be reconstructed in the correct place, and in which situations objects were relocated. This is the basis of figures 3.28 to 3.35.

Each of the four cases (each of which has two figures related to it) started off with a prescribed background density. Then an object of fissile material of size equal to one voxel was placed voxel number 1 and the data relating to it was calculated. Next, the first step of the OMP was enacted, using a fixed smooth density distribution for $\boldsymbol{\rho}_2$ (not necessarily the correct one) and an assumed sparsity level of 1 for the sparse distribution $\boldsymbol{\rho}_1$. Based on the data created we could find out where the first sparse reconstruction could occur during OMP. This was then repeated for all voxels within the cargo container to determine what areas the fissile material could be detected under different conditions.

There are two columns of figures, each focusing on only one cross sectional slice of the cargo container. Those on the left portray the regions where the fissile material is found in the correct location (black) and voxels which are moved to a different location (white). Those on the right generally deal with where the fissile material is relocated to. Some regions of the cargo container (black) never contain reconstructed fissile material whereas others (white) obtain it in at least one instance.

Figures 3.28 and 3.29 cover the case where the actual density distribution is a constant background density equal to 1g/cm^3, and the starting point is the same. So

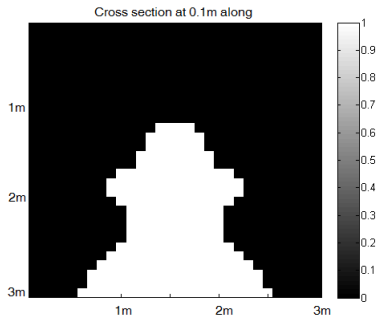

Figure 3.28: White means the voxel of fissile material is found in the wrong location. The actual background distribution is a smooth background of density 1g/cm^3. Starting point is the same.

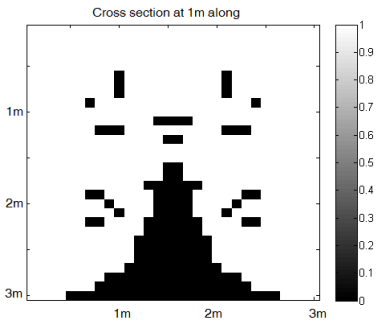

Figure 3.29: White means the voxel of fissile material has been moved here from its actual location. The actual background distribution is a smooth background of density 1g/cm^3. Starting point is the same.

in this case we are assuming the genetic algorithm has found the exact solution for $\boldsymbol{\rho}_2$ before we enact the OMP for $\boldsymbol{\rho}_1$.

Figure 3.28 shows a pattern we should expect - fissile material towards the center of the cargo container will be harder to correctly locate due to the diminishing intensity of gravity gradiometry, and similarly for material on the lower edge due to a lack of sensors there. Even under such perfectly prescribed conditions this is the case.

It would be difficult (or at least confusing) to attempt to display where each of the voxels coloured in white were reconstructed, thereby tracking the movement of such voxels in the reconstruction. Instead only one voxel was tracked - the voxel with the lowest number assigned to it that was not reconstructed in the correct location. It appeared in the cross section at 1m along, which is why that slice was chosen for figure 3.29. The reason for this slice being the moving point is most likely due to the locations of the sensors, as this slice is very close (if not the closest) to the sensor gate. However even in this slice there are regions (in black) where no fissile material is ever reconstructed, and in fact it is easier to see the general sphere of detection for the sensors by noting the curve of the boundary between the two colours as we approach the lower edge of the cargo container. It is also fascinating that there are small regions of black that cannot be described as being too far from the sensors. Such regions occur slightly more towards the corners and so there could be some interaction between different sensors which might cause this.

However that is under very good conditions, whereas figures 3.30 and 3.31 change

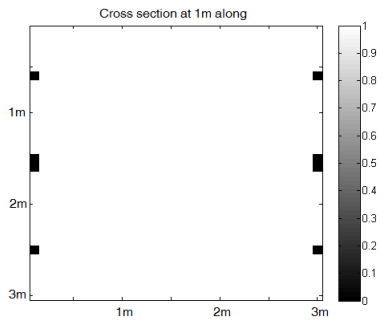

Figure 3.30: White means the voxel of fissile material is found in the wrong location. The actual background distribution is a constant density of 1g/cm^3. Starting point is the distribution of zeros.

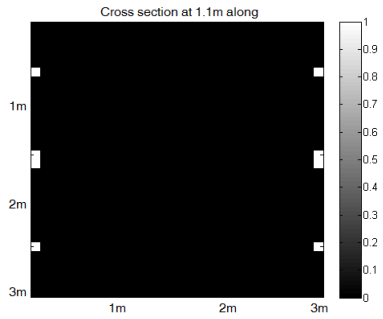

Figure 3.31: White means the voxel of fissile material has been moved here from its actual location. The actual background distribution is a constant density of 1g/cm^3. Starting point is the distribution of zeros.

them somewhat, in that everything is the same except the starting position before entering the OMP. Instead the smooth reconstruction $\boldsymbol{\rho}_2$ is made up of zero density, which is akin to not enacting the genetic algorithm at all and assigning a trivial initial position. The differences are drastic as figure 3.30 shows the first voxels at which the fissile material is correctly located, and this is in the cross section at 1m along which is close to the sensors. Any object beyond approximately 1 voxel away from the sensors is not correctly located after 1 iteration of the OMP. Combine this with the results from figure 3.31 and we find that all such voxels are relocated next to the sensors, no matter how far away they are. A good starting point for the OMP has a huge impact on the results, which makes the work of the genetic algorithm even more important.

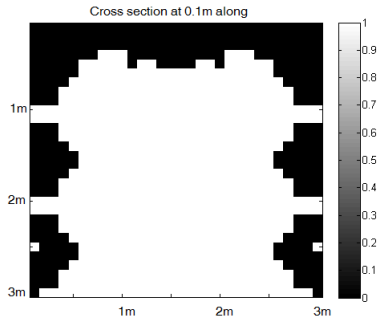

Figure 3.32: White means the voxel of fissile material is found in the wrong location. The actual background distribution is one representing stacked pallets. Starting point is the same.

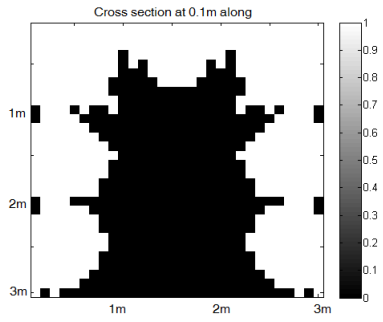

Figure 3.33: White means the voxel of fissile material has been moved here from its actual location. The actual background distribution is one representing stacked pallets. Starting point is the same.

Figures 3.28 and 3.29 showed us a simple example of when the genetic algorithm had achieved perfect results, and so figures 3.32 and 3.33 look at a more cluttered case. Here the background density is that of stacked pallets shown in figure 3.10, and the starting point is the same. Comparing 3.32 to 3.28 we see the radius of detection of the sensors (denoted in black) has been reduced, which is unsurprising given the increase in complexity. According to figure 3.33 it seems the regions where no voxels are ever found has also increased. If we assume that the genetic algorithm can correctly achieve such a realistic example, the OMP fails to find the fissile material in the correct location for most of the domain.

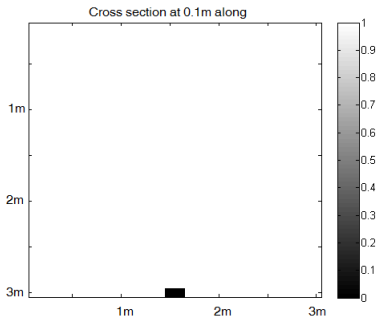

Figure 3.34: White means the voxel of fissile material is found in the wrong location. The actual background distribution is one representing stacked pallets. Starting point is the distribution of zeros. A weighting function is also used.

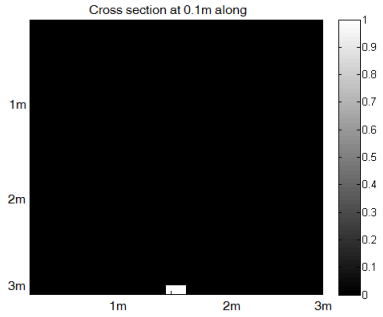

Figure 3.35: White means the voxel of fissile material has been moved here from its actual location. The actual background distribution is one representing stacked pallets. Starting point is the distribution of zeros. A weighting function is also used.

The background density for figures 3.34 and 3.35 is the same as that for 3.32 and 3.33 - stacked pallets. The approach is similar to that followed by 3.30 and 3.31 - a starting point of zero density. However this alone would only achieve a repeat of figures 3.30 and 3.31. Instead we employ a weighting function within the OMP, one previously seen which uses the distance from the sensors in order to counteract the degradation of the signal, thereby allowing objects further from the sensors to be reconstructed. Unfortunately this results in an overcompensation as shown in figures 3.34 and 3.35. All objects are reconstructed on the lower edge of the cargo container, as far away from the sensors as possible, which fails in the same way as before.

This section has shown that there are some failings of looking for a sparse reconstruction, one being that it relies heavily on the solution of the genetic algorithm.

However we can't know how accurate a solution the genetic algorithm can give us, given that the ability to recreate more complex density distributions relies on the number of radial basis functions used. For example, for some of the results seen so far 1000 iterations of the genetic algorithm were enacted before looking at sparsity. It was run for 10000 iterations but the results were very similar. It's possible that by using more radial basis functions in fixed locations we can arrive at a better solution, but after a certain point it will take too long to use that many RBFs.

Sparsity may not be a valid option in its current form, but that does not mean an alternative approach will not benefit from its use. A more forceful two stage process could be the way to go. Sparsity may simply require a more restricted domain to work on. It was found in [51] that the reconstruction was smoothed out, and so a threshold value had to be found based on that by use of funnel analysis. However it is possible that a sparse-based processing technique could instead be utilised on those regions of higher density. The analysis performed in this chapter should outline the need for the careful application of this technique, but I think there are avenues with which this could be effectively applied.

Chapter 4

Incorporating radial basis functions into the colour level set scheme with gradient descent

In the previous chapter we introduced the use of radial basis functions for the purposes of speed in the genetic algorithm. However we haven't yet addressed the question of whether such functions can be useful in the gradient descent scheme. It may provide some improvement on the results we have seen so far. Radial basis functions have been used before in the parameterisation of level set functions [35] and [81] and they should be a good fit.

Let us begin by redefining how the domains depend on the level set functions. The motivations behind this change will be explained during this chapter.

The domains are defined as follows

- $D_1 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) > 0 \text{ and } \phi_2(\boldsymbol{x}) > 0 \text{ and } \phi_3(\boldsymbol{x}) > 0\}$
- $D_2 = \{\boldsymbol{x} : \phi_1(\boldsymbol{x}) \leq 0 \text{ and } \phi_2(\boldsymbol{x}) > 0 \text{ and } \phi_3(\boldsymbol{x}) > 0\}$
- $D_3 = \{\boldsymbol{x} : \phi_2(\boldsymbol{x}) \leq 0 \text{ and } \phi_3(\boldsymbol{x}) > 0\}$
- $D_4 = \{\boldsymbol{x} : \phi_3(\boldsymbol{x}) \leq 0\}$.

4.1 Radial basis functions

As before we are looking at each level set function being made up of Gaussian functions of the form

$$f_k(\boldsymbol{x}) = a_k \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{b}_k||.^2}{2c_k.^2}\right). \tag{4.1}$$

In this case a_k designates the magnitude of the Gaussian function, $\boldsymbol{b}_k$ represents the coordinates of the center and c_k controls how far-reaching it is. Previously all three have been unknown, but now we fix both $\boldsymbol{b}_k$ and c_k so that a_k is the only unknown. On the surface this would seem to speed up the algorithm as we have fewer unknowns. However since these Gaussian functions cannot move around the domain and they have restricted reach we will need to have enough to completely cover the domain. So we cannot say for sure whether there would be an increase in speed. Also we make a further alteration and instead use truncated Gaussian functions

$$g(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if} \quad f(\boldsymbol{x}) > \epsilon \\ 0 & \text{if} \quad f(\boldsymbol{x}) \leq \epsilon \end{cases} \tag{4.2}$$

where $\epsilon \ll 1$ is a prescribed tolerance.

Fixing the centers has some advantages though is briefly explored in a previous section. Previously we have had some trouble with the high density material being reconstructed closer to the sensors than it should be. If the Gaussian functions are a certain distance away from the sensors then this can be prevented somewhat. Since there is only one level set function controlling the lead then we can exert more control over this aspect.

By defining a uniform mesh of radial basis functions we can cover the whole cargo container. One idea would be to make said mesh adaptive - start off with a coarse mesh and run an algorithm for a set number of iterations, or until it has sufficiently converged. Then seek out the regions of higher density (say domains D_3 and D_4) and refine the mesh in those regions before carrying on. Thus we would obtain a higher resolution in the regions with higher density, and a lower resolution in less interesting regions with lower densities.

Unfortunately this course of action slows down the algorithm due to the increase in unknowns as the method progresses. Certain choices can be made to alleviate this problem. For instance each Gaussian function is non-zero in a relatively small region

and so sparse matrices could be utilised for speed and storage if required.

Since we are using a non-adaptive mesh we need to choose the parameters involved as carefully as possible. The fixed centers $\boldsymbol{b}_k$ give us a rough idea on the resolution we are able to reach - we know we can resolve an object of size equal to the distance between two radial basis functions, and objects smaller in size depending on the magnitudes of a_k for respective radial basis functions. Obviously a finer mesh leads to better resolution but there is still the question of speed.

The other parameter to choose is c_k. This performs the same duties as the standard deviation parameter in a Gaussian distribution (where $a_k = 1$) and so must be chosen carefully. Here we link it to the tolerance ϵ introduced above. Since the function will become zero when it reaches that tolerance we require a value of c_k which allows each function to reach the neighbouring one, thereby achieving complete coverage of the domain.

If we set $a_k = 1$ (as this is the maximum) and let h be the distance between radial basis functions then a plausible choice would be

$$c_k = -\frac{h^2}{2\log\epsilon}. \tag{4.3}$$

In one dimension this may be sufficient, but not when we extend to three dimensions. Instead this c_k is treated as a starting point which is then increased until full coverage is achieved.

By fixing b_k and c_k the reliance is completely on α_k to vary the size of recovered objects. b_k are fixed first and c_k are found such that the radial basis functions overlap, and so there may be a prevalence towards reconstructing larger objects, while smaller objects are harder to recover. However the only domain where smaller objects may be required is $\mathcal{D}_4$, which is controlled by a single level set function being negative.

Each radial basis function is accompanied by 8 other radial basis functions in its immediate vicinity. For simplicity let us set $\alpha_k = 1$ for all such 8 so that the only parameter to be aware of is α_j for the middle radial basis function. Setting this to $\alpha_j = -1$ is one extreme - creating an object which reaches roughly half the distance to each of the neighbouring radial basis functions (the partial overlap of the other 8 may create a cumulative positive effect).

Increasing α_j then decreases the size of the object representing $\mathcal{D}_4$. However this will have a limit and there will be a reliance on the distance between each radial basis

function - essentially the coarseness of the radial basis function mesh, with each mesh point being the center of a radial basis function. This will also tie into the mesh established by the voxels. Generally speaking at least one voxel is left remaining for each domain - otherwise there is no boundary left for the level set method to shape. Therefore the aim would be the ability to reconstruct a single voxel.

During this chapter a relatively coarse mesh is employed, which will most likely have a large subspace error associated with it. However the purpose of this chapter is not necessarily to obtain satisfactory reconstructions by this method alone - instead it is meant to be a stepping stone for the next chapter, performing the required analysis on this approach before incorporating it into a larger scheme.

(As a side note the next chapter uses a finer mesh - the cross section of the cargo container contains a 5×5 grid of radial basis functions, with 11 extending along the length of the cargo container. Using the simple examples as a guide and a lens of 20cm $\times$20cm $\times$20cm as a guide, it was found that some examples the average density was slightly less than that of lead. This means for certain solutions created in the ensemble algorithm the largest object of domain $\mathcal{D}_4$ was less than the size of the lens - which equated to $2 \times 2 \times 2$ voxels, which may be the subspace error for that set up.)

The aim is to combine the genetic algorithm with a method of steepest descent, but they are not directly compatible with each other. Choices were made to make the combination easier and smoother. For instance the steepest descent method is easiest to apply to a smooth continuous function. This motivates the use of a level set function as it provides the mapping from continuous to discontinuous domains.

The genetic algorithm does not require a continuous function to work on - it could incorporate discontinuities without it. However the genetic algorithm does need to communicate its solutions to the steepest descent method, and so level set functions are used again. Unfortunately the genetic algorithm is very slow, and so the unknowns are reduced by means of a reparameterisation. Bearing in mind the smooth nature of the level set functions it seemed that the best option would be to choose smooth basis functions. Hence why these radial basis functions are used. It may be there exists a better approach which still allows for easy communication between the two methods.

4.2 Gradient descent on radial basis functions

Using the above domains and setting the density to be ρ_i for domain D_i we have the following formulation

$$\begin{aligned}\rho(\boldsymbol{x}) &= \rho_1 H(\phi_1(\boldsymbol{x}))H(\phi_2(\boldsymbol{x}))H(\phi_3(\boldsymbol{x})) && (4.4)\\ &+ \rho_2(1-H(\phi_1(\boldsymbol{x})))H(\phi_2(\boldsymbol{x}))H(\phi(\boldsymbol{x})) && (4.5)\\ &+ \rho_3(1-H(\phi_2(\boldsymbol{x})))H(\phi_3(\boldsymbol{x})) && (4.6)\\ &+ \rho_4(1-H(\phi_3(\boldsymbol{x})). && (4.7)\end{aligned}$$

The Heaviside function is equal to 1 in positive regions and 0 in negative regions, but we use a continuous approximation

$$H_\epsilon = \begin{cases} 0 \quad \text{if} \quad x < -\epsilon \\ \frac{1}{2}(1+\frac{x}{\epsilon}+\frac{1}{\pi}\sin(\frac{\pi x}{\epsilon})) \quad \text{if} -\epsilon \leq x < \epsilon \\ 1 \quad \text{if} \quad x \geq \epsilon \end{cases} \tag{4.8}$$

which can be found in papers such as [3] and [48]. Expanding on the format previously described we can define the gradient descent direction for the unknown parameters to be

$$\frac{da_k}{dt} = -G^T(G\rho - d)\frac{\partial \rho}{\partial \phi_i}\frac{\partial \phi_i}{\partial a_k} \tag{4.9}$$

where the ϕ_i referenced is the level set function which utilises the specific a_k we are changing. Also the differentials with respect to each level set function are, now that we have redefined the domains,

$$\frac{\partial \rho}{\partial \phi_1} = (\rho_1 - \rho_2)H(\phi_2(\boldsymbol{x}))H(\phi_3(\boldsymbol{x}))\delta_\epsilon(\phi_1(\boldsymbol{x})) \tag{4.10}$$

$$\frac{\partial \rho}{\partial \phi_2} = ((\rho_1 - \rho_2)H(\phi_1(\boldsymbol{x})) + \rho_2 - \rho_3)H(\phi_3(\boldsymbol{x}))\delta_\epsilon(\phi_2(\boldsymbol{x})) \tag{4.11}$$

$$\frac{\partial \rho}{\partial \phi_3} = (((\rho_1 - \rho_2)H(\phi_1(\boldsymbol{x})) + \rho_2 - \rho_3)H(\phi_2(\boldsymbol{x})) + \rho_3 - \rho_4)\delta_\epsilon(\phi_3(\boldsymbol{x})) \tag{4.12}$$

where

$$\delta_\epsilon = \begin{cases} 0 \quad \text{if} \quad x < -\epsilon \\ \frac{1}{2}(1+\frac{x}{\epsilon}+\frac{1}{\pi}\sin(\frac{\pi x}{\epsilon})) \quad \text{if} -\epsilon \leq x < \epsilon \\ 0 \quad \text{if} \quad x \geq \epsilon \end{cases} \tag{4.13}$$

is the differential of the approximate Heaviside function. As before this keeps the update centered around a small strip around the boundary, however the final term in the gradient descent algorithm can change that slightly as

$$\frac{\partial \phi_i}{\partial a_k} = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{b}_k||^2}{2c_k^2}\right) \tag{4.14}$$

for a specific a_k used to build ϕ_i. Under very specific circumstances (when a drastic change is required) the whole region of the domain that this radial basis function reaches could potentially change, but it is very unlikely. So the above step equates to increasing the size of the narrow band around the boundary of the level set function at certain points in the algorithm.

4.2.1 Initialising the parameters

Each parameter a_k is chosen to reside in the interval $(-1, 1)$, and the other fixed parameters have been calculated based on that assumption. Given that we won't know much about the inside of the cargo container it is unlikely we can justify any starting point as being feasible, without unnecessarily influencing the final result. A random starting point would be the most defendable position to be in, as this gives us complete freedom to reach any plausible solution that matches the results.

However there may be consequences to this due to the restrictions already imposed. Hypothetically speaking say one of the unknown parameters starts near the boundaries of -1 or 1. If the gradient descent direction points outside the boundary then we are restricted in the step length we are able to use - a smaller step length will have to be settled on.

Parts of the code have been altered in case this happens by simply not allowing any offending parameter to exceed (or be equal to) the interval boundaries. However it would be preferable that it only happened later in the algorithm, as allowing a very positive or very negative basis function to remain from iteration to iteration could influence the solution reached and we would rather the early stages have more freedom. This motivates the decision to choose the parameters in the smaller interval of $(-0.1, 0.1)$ instead when starting the algorithm.

4.2.2 Line search employed

A lot of work has been performed on the line search method in previous sections. Such methods can be problematic when missing key information about the problem at hand, for instance when we have no information on how large a step size to expect. Radial basis functions can help us here - since the parameter being optimised is restricted to a certain interval that gives us a good starting guess as to how large a step size is allowed (with some extension if needs be).

However simply knowing our limits may not be good enough if the cost functional is increased when using the maximum step size. Then we have to backtrack and find a useable step size in an interval which might be relatively large, with a matrix-vector multiplication at each step. Instead we can use more of an approximate line search, one in which the aim is not necessarily to reduce the cost functional at each step, but aim for a general downward trend over several steps. To do this we track instead how many voxels are changing at each iteration.

The method is as follows. At each steepest descent iteration we have 3 level set functions ϕ_i, for $i = 1, 2, 3$ to deform, each with their corresponding gradient direction $\delta\phi_i$ respectively. Assign to each one an integer v_i which corresponds to a number of voxels. We then find τ_i such that the number of voxels changed when this step length is applied is less than v_i. Given a suitable v_i this should generally decrease the cost functional without being computationally expensive and with the added bonus of possibly reducing the chance of simply getting closer to a local minimum point. Instead it may circle the minimum point, or even break free if that is achievable. One other advantage is that it is more likely to provide the genetic algorithm with a cluster of points near to a minimum rather than a short path pointing to it. Also there will be only one matrix-vector multiplication per iteration when calculating the cost functional.

The main problem is to calculate v_i. When we begin to enact the gradient descent method the first iteration deals with this. There we look for τ_i such that the cost functional decreases, without necessarily satisfying the Wolfe conditions. This means that the cost functional should decrease at least once during the gradient descent direction. After having applied each τ_i separately we then count how many voxels have changed for each one, which we call $\hat{v}_i$. Then $v_i = \alpha_i \hat{v}_i$ where $\alpha_i \in (0, 1]$.

During the first iteration the search for a decrease in the cost functional requires a few distinct steps. First we find the maximum allowable step length τ_{max} and record the cost functional $\mathcal{J}_{max}$ associated with it. If this step length has increased the cost functional then it is reasonable to assume we have gone too far and missed the minimum point in this direction and so we must reduce τ_{max}.

Our next step is to perform a bisection step and obtain $\tau_{mid} = 0.5\tau_{max}$ along with its associated cost functional $\mathcal{J}_{mid}$. Then if this new cost functional does not satisfy our conditions we go further by using the three points we have to form a quadratic linking the step length and cost functional then calculating the minimum point. Note that we could have calculated a quadratic using the two points plus the gradient at $\tau = 0$ but a bisection step may be enough to satisfy the condition and so it is employed first in an attempt to save on time.

In subsequent iterations we look for a value of τ which achieves the number of voxels changed to be less than v_i by employing a form of backtracking. It is relatively simple to find a maximum value for τ based on the restrictions of the radial basis functions. If this τ changes more than the allowable number of voxels then we need to reduce it. Given that we know a step length of 0 results in no voxels changing we can easily calculate a proposed linear relationship between τ and the number of voxels changed. Using this we can find an approximate value for τ which should provide us with the number of voxels changing being equal to v_i, then take a proportion of said τ in order to more reliably reach below v_i. If we were to assume the calculated step length was correct then we may instead end up converging to v_i from above but never actually reaching it as we cannot assume the linear relationship is correct. By performing subsequent iterations we should find a non-zero τ which satisfies the conditions.

Sometimes this v_i might not be the best choice as the cost functional is not decreasing at all. If it increases twice in a row then we can safely assume that v_i is too high and therefore decrease it. Otherwise we will continue to miss the minimum point. Similarly, it is possible for the method to oscillate between two cost functionals, and so v_i is reduced in such a situation in order to reduce the chances of oscillation. Also a finite number of iterations are used in the steepest descent method before a step length of zero is returned to save on computational time. At that point we can

assume v_i may be unreachable under the current situation and therefore increase v_i to accommodate it.

4.2.3 Gradient descent with RBFs results

Below is a single example where one small lead object is present in the domain. Elsewhere within that domain there exists one metal box and one wooden box but only the location for the high density material is shown in figure 4.1. After enacting the algorithm outlined above the solution found is shown beside it in figure 4.2, and unfortunately it is not very promising. The small lead object has moved to the edge and increased in size, and other lead objects have been reconstructed elsewhere in the domain.

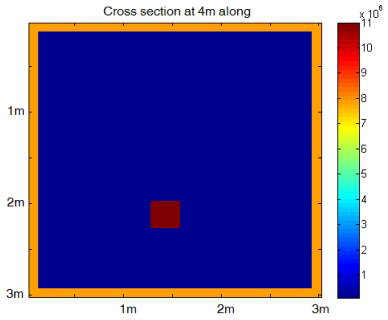

Figure 4.1: Cross section of the cargo container containing the small lead object.

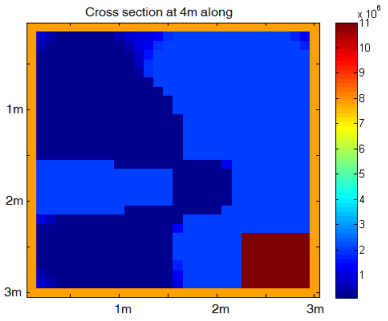

Figure 4.2: Reconstruction of 4.1. Only one slice is shown.

In order to find out what has happened during the running of this algorithm the cost functional has been plotted in figure 4.3 for the first 20 iterations. It begins rather well, with some fast convergence in the first 5 iterations, but then slows down drastically. Due to the relaxing of the reduction in the cost functional it is now able to oscillate, accompanied by a general downward trend.

Figure 4.4 shows the cost functional for iterations 21 to 100 and it has begun to exhibit more of an oscillatory motion and less of a downward one. Towards the end of the first 100 iterations the range it is covering has begun to shrink, indicating a possible convergence. This is confirmed in figure 4.5 where there is hardly any reduction in the cost functional and it is stuck in the region of $\mathcal{J} = 13.2742$, which is not the lowest cost functional achieved as that was reached within the first 100 iterations.

The reason behind this behaviour is not a failing in the method but rather a result

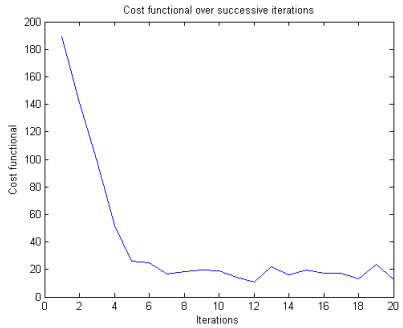

Figure 4.3: Cost functional for the first 20 iterations.

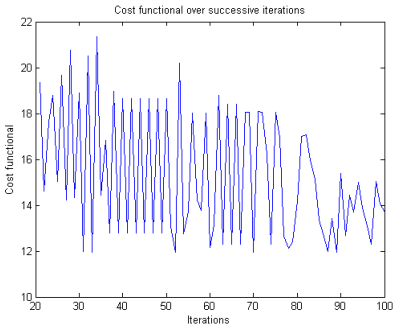

Figure 4.4: Cost functional for iterations 21 to 100.

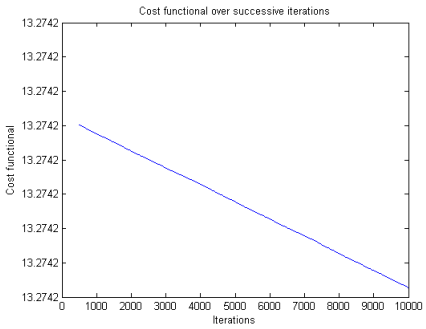

Figure 4.5: Cost functional for iterations 500 to 10000.

of the restrictions placed on the density distribution. Since it relies on radial basis functions, which in turn are based on the parameter a, there is a limit to the changes that can be made. This is because a has been restricted to the interval of $(-1, 1)$ which in turn restricts the size of the step length we are able to use.

Alterations to the method were explored in order to get around such an event. This involves simply identifying any parameter a which might be restricting the evolution and keeping it separate from the line search method. However early results suggested that such a workaround might be in vain. It is possible that a better method could be developed but this wasn't pursued. Instead another use was envisioned for this form of the method.

We can take two main insights from the results above. The first is that fast convergence in such situations where the algorithm has room to move, but usually begins to falter after 20 or so iterations. The second is that even when it is not converging it is good at exploring the surrounding area near to the local minima imposed by the current parameterisation. These two aspects motivate its use in conjunction with another method, which is the basis of the next chapter.

Chapter 5

Use of an ensemble hybrid scheme

5.1 Taking the best of both worlds

Both the genetic algorithm and steepest descent approach have their strengths and failings. The steepest descent algorithm relies on a gradient direction to work sufficiently well. Since the cost functional in our problem gets relatively flat in later stages it can be difficult to obtain a good decrease. There is a possibility of getting stuck near a certain solution with little chance of escaping (a possible local minimum). Alternatively the genetic algorithm aims at more global convergence - it is less likely to get stuck near a local minimum. However it could take a while to find a better fit. It would be advantageous to find a way to combine the two methods into one, making use of the strengths of both.

5.1.1 Combining the two methods

A genetic algorithm views a single density distribution as an individual, with many individuals making a population.It requires the use of such a population of individuals to work on, whereas the steepest descent approach only requires a single individual. It is therefore necessary to come up with a way to transition between the two structures. For instance, during any gradient run we follow a path through the iterations, sampling a solution at prescribed steps, thereby building a small population for the genetic algorithm to work on.

However this provides us with a limited region of the domain. Using a second

individual in the gradient descent algorithm will give us a greater exploration of the parameter space. It might be that the second individual converges to a similar point, but the path used to reach it will be different and one of the requirements of the genetic algorithm is an adherence to a minimum amount of variability within the population it is working on. Therefore there will be a small population of individuals for the steepest descent algorithm to work on, and sampling throughout the method provides the genetic algorithm with a larger and more varied population to work on.

That is one switching point (from gradient to genetic) now we have to look at the other. At the end of the genetic algorithm we have a large population that we need to sample from, the first obvious choice being the individual with the best fit. The second choice may also be relatively obvious as it is the individual with the second best fit. This may be very similar to the best fit individual, or it might be radically different and yet still provide us with a relatively good fit. Either possibility fits the description of a worthy competitor in the genetic algorithm. The rest of the individuals are chosen at random regardless of fit. The reason for this is that even though they might not currently be good options, through following the path of steepest descent they might become competitors later on as the steepest descent algorithm is capable of fast convergence over a few steps. Here we have combined the strength of the genetic algorithm retaining less able individuals for variability reasons with the strength of the steepest descent algorithm in being able to home in on local minima. The line search method used in the steepest descent algorithm is that explored in the previous chapter. One advantage of this line search is that it is more likely to provide the genetic algorithm with a cluster of points near to a minimum rather than a short path pointing to it.

5.1.2 Expected errors

Due to the aim of avoiding an inverse crime the finite element mesh we use to reconstruct the density distribution is not the same as the one used to create the data. A finer mesh is used for the data in order to model the expected disconnect between a discrete and continuous domain. Therefore we cannot assume to reach a cost functional equal to zero, or upon reaching zero we might not have a correct reconstruction, thus an expected aim for the cost functional needs to be found.

As we reach a certain value of the cost functional it becomes increasingly difficult to reduce it even further. However we can't find a value based on the behaviour of the algorithm - the steepest descent method becomes very slow in the later stages of the algorithm anyway and the genetic algorithm might not obtain a better fit for an elongated piece of time, neither providing us with the indicators of convergence we require. Therefore the tolerance must be decided upon and prescribed before the algorithm begins.

Finding a feasible tolerance itself is an optimisation problem. If we prescribe it too low then the algorithm might never reach it. Too high and the solutions obtained might not be the best we can obtain. We begin the only way we can at this stage - attempting to find it experimentally. First we take a test problem, such as a simple one shown below in figure 5.1. One slice of the cargo container is shown which is the one containing the lead object. Since this is a simple example there are very few objects in the cargo container. In the slice shown only lead is present as a cube of 25cm on each side. There are only two other objects within the container, one being a wooden cube 1m on each side and the other being a metal cube 50cm on each side. This is meant to be the simplest example.

Also included are two slightly more complex examples - both have multiple objects of varying sizes made of wood and metal in the background to attempt to hide the lead object. The locations of all objects were chosen at random so no form of structure for these cargo containers has been assumed. However in figure 5.2 the lead object is being shielded slightly by a metal object between itself and the wall. Also a larger metal object is present nearby, which could confuse the method. Not shown here is that elsewhere in the cargo container has a high concentration of wooden objects. If we were to equate this to a real world example it could be that some higher density objects have been hidden at one end of the cargo container, possibly at the end not containing the door in an effort to smuggle the object through any checking system.

In figure 5.3 the small lead object is separate from any metal objects, but there are a couple nearby so a similar problem may occur as before. There is a lower concentration of wooden objects in this example compared to the previous cluttered one so that will be a good comparison. For the purposes of clarity during this section the three examples will be referred to as the simple example, the first cluttered example

and the second cluttered example respectively.

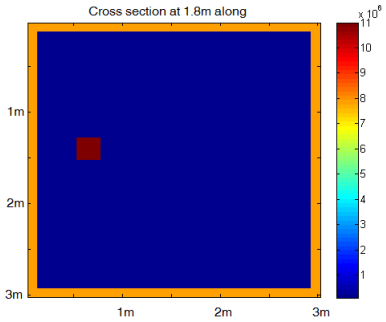

Figure 5.1: Cross section of the cargo container containing the small lead object. This is an uncluttered example.

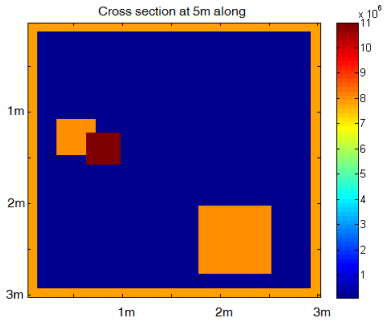

Figure 5.2: Cross section of the cargo container containing the small lead object. This is the first cluttered example.

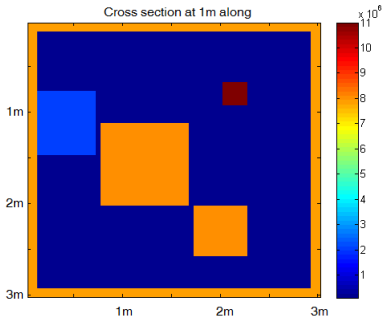

Figure 5.3: Cross section of the cargo container containing the small lead object. This is the second cluttered example.

Next we create a batch of what we deem 'correct' solutions. We begin with the actual density distribution on the finer mesh created using what we call data voxels. Using this we create the same density distribution on the reconstruction voxels. Note that this might not be an accurate match as the examples were chosen without the reconstruction mesh in mind in order to create more of a mismatch between the data and reconstruction.

For instance it is not possible to correctly model an object of size 0.25m on each side (figure 5.1) using voxels of size 0.1m as used in the reconstruction. In this case two separate density distributions are created - one with an object of size 0.2m and one with 0.3. However if the actual object was of size 0.2m then there would be three possible density distributions created - one with an object of size 0.2m along with objects of size 0.1m and 0.3m. This is to allow for slight variations in the reconstruction.

This is also not restricted to the lead object alone - all objects are allowed the same slight change to their size. Also the location is allowed to deviate slightly from the correct place - up to a maximum of one voxel in each coordinate direction. By creating a batch of density distributions made up of all possible permutations of sizes and locations of each object we can calculate a range of cost functionals that would indicate a correct solution.

The batch of solutions is meant to be a starting point - this is something we would not have access to when presented with a cargo container with unknown contents. Later an attempt will be made to link it to the weight of the cargo container, but it is still treated as an approximation that is allowed to adjust itself if found lacking. It is likely that as the cargo containers become more realistic as opposed to these simple examples such an approximation will begin to degrade and so an alternative will have to be sought, perhaps by studying the behaviour of the ensemble method as it approaches a solution. For now this approximation is utilised throughout this chapter.

The actual solution is prescribed directly onto the voxels, bypassing the radial basis functions, therefore this may not be achievable for coarser RBF meshes. The assumption is that we use a fine enough mesh to reach this solution accurately. Using the above three examples we obtain a range of cost functionals grouped together in histograms as shown in figures 5.4 to 5.6 for figures 5.1 to 5.3 respectively.

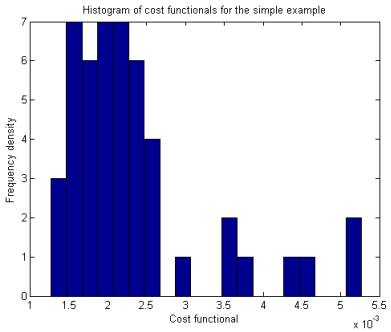

Figure 5.4: The range of cost functionals for the simple example that we would consider allowable.

Each example creates a certain range of cost functionals. As we reach that range during the method we can be reasonably sure we have something close to an accurate solution. Unfortunately said ranges can fluctuate due to unknown factors. For example

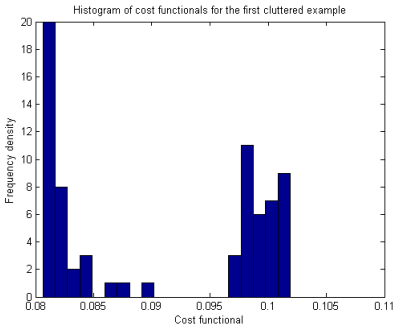

Figure 5.5: The range of cost functionals for the first cluttered example that we would consider allowable.

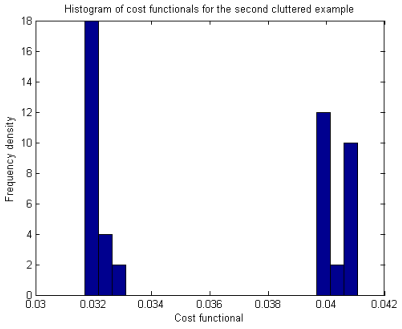

Figure 5.6: The range of cost functionals for the second cluttered example that we would consider allowable.

if the fissile material is close to a sensor then a larger cost functional could be a correct solution, whereas those further away from a sensor require a lower cost functional to be considered correct. However we won't have access to this information before we open the cargo container.

Similarly the number of objects within the cargo container also changes the range. A more cluttered example does not require a low cost functional to reach a solution, and in fact we might not be able to reach lower cost functionals in such cases. However this is still something we won't know beforehand. What we will know is the weight of the cargo container. For each example we look at the maximum plausible cost functional - the value at which we have entered the region of plausible solutions. In figure 5.7 below the recorded maximum is plotted against the cargo container weight, resulting in a general trend of positive correlation. By using a line of best fit (the built in MATLAB operator) we come up with an approximate expected error.

Now we cannot use this as a strict tolerance value for two main reasons. First is that it is found using empirical evidence and the examples vary from the line of

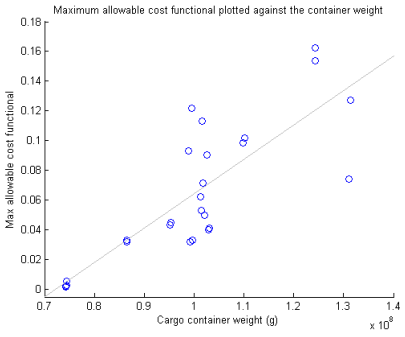

Figure 5.7: The maximum allowable cost functional plotted against the cargo container weight.

best fit, so the actual tolerance could vary. Secondly this calculated value is simply a maximum value. As we reach it we can only be slightly certain that we are approaching a good solution. There could be other reconstructions with similar cost functionals that are not as close to the actual density distribution. What we do know is that other plausible solutions can exist below this value, and so we need to allow the method to access them.

Let us say that the calculated tolerance is lower than what the algorithm is able to easily reach. The outcome could be it takes longer to reach a solution or it cannot reach a solution at all. Even if it does reach a solution then we have discounted any solution higher than the imposed tolerance without reason. This motivates us to adjust the calculated tolerance in some way depending on what is currently occurring within the method.

As stated several times the steepest descent algorithm will slow down in the later stages of the method, and due to the first step will likely still be reducing the cost functional slightly so it would be difficult to use this as an indicator. Our only other option is the genetic algorithm. This can spend many iterations without improving the best fit, but the longer it runs the more likely it should find a better fit. Our proposal is that if it completes one cycle (i.e. the genetic algorithm is run for a prescribed iterations before preparing to run the steepest descent algorithm again) without reducing the cost functional then we can assume it is possibly stuck.

In such a situation we slightly increase the prescribed tolerance before running the next cycle, thereby decreasing the chances of getting stuck at a certain point. This may seem drastic in principle but may not occur often in practice.

The reason behind this is due to a byproduct of one of the strengths of the genetic algorithm. Having the mutation constant based on the variation of the population we are always attempting to keep some variance to allow a more global convergence. However this variability is tracked using the measure of variety [22], which works on the best fit and the median fit. As long as the median fitness is relatively far from the best fit then the variability can be described as high. However this will allow for a smaller group of individuals to be very close to the best fit without affecting the variability and therefore mutation parameter.

Such individuals would be more likely to pass on their information to the next iteration, but would be limited in size. As their population increases they become numerous enough to affect the median fitness, consequently increasing the mutation parameter and therefore reducing their numbers. Let us assume that a similar cost functional implies a similar density distribution (as at least some of them would be similar). Over many crossover stages it becomes more likely that two individuals from this sub-population will find themselves in the same crossover point, leading to offspring that are very similar to both parents and an evolution closer to that of the gradient descent algorithm. Therefore within the genetic algorithm there could be a form of local minimisation happening, and over may iterations one offspring may be created that is slightly better than both parents. Hence the alteration proposed will only be activated in dire situations.

In order to address the second problem we assign an individual a probability of being accepted as it reaches below this threshold. The probability would be small at first, and then increase as it decreases further. Let us define the calculated tolerance as $\mathcal{J}_{tol}$ and our current cost functional $\mathcal{J}$. We begin by defining a distance measure

$$\mathcal{J}_d = \frac{\mathcal{J}_{tol} - \mathcal{J}}{\mathcal{J}_{tol}}. \tag{5.1}$$

This satisfies our requirement of becoming more positive as $\mathcal{J}$ decreases below $\mathcal{J}_{tol}$, and it converges to 1 as $\mathcal{J} \to 0$, therefore giving us a probability of being accepted, which has some parallels with Markov Chain Monte Carlo methods. However since the genetic algorithm can take some time to change the best value, there could be several iterations where $\mathcal{J}$ remains very close to $\mathcal{J}_{tol}$, and repeated probability tests increase the chances of being accepted, resulting in more solutions being accepted close to $\mathcal{J}_{tol}$. As mentioned before, $\mathcal{J}_{tol}$ has been chosen to represent more of the outliers at the

higher end of the accepted cost functionals and we want to have solutions shifted more towards the negative end. Therefore the acceptance probability used is

$$A_c = \mathcal{J}_d^2. \tag{5.2}$$

If the tolerance has been chosen relatively close to the correct acceptance tolerance then the above probability should give us a distribution closer to those seen above. This therefore relies on the first alteration to the code working correctly, which assumed that if the genetic algorithm has stopped it is safe to assume we need to increase the acceptance tolerance. However using the above probability it is possible to be below the acceptance tolerance but still have the genetic algorithm not converging.

For instance let $\mathcal{J}_{min}$ be the current minimum cost functional for the population of individuals within the genetic algorithm. If $\mathcal{J}_{min}$ remains constant throughout the genetic algorithm then it is possible that the genetic algorithm has become stuck and will never reach the prescribed tolerance. In such a case it is assumed that the tolerance is too small and so $\mathcal{J}_{tol}$ is increased slightly.

However if $\mathcal{J}_{min} < \mathcal{J}_{tol}$ then there is no need to increase the tolerance as the algorithm has reached the region of possible solutions. Therefore the tolerance is only increased if $\mathcal{J}_{min} > \mathcal{J}_{tol}$.

5.1.3 Creating a population of solutions

Using the above method will only give us a feasibly good solution. To improve our odds of finding a correct solution we run the algorithm until we find a prescribed population of solutions, thereby giving us a percentage of solutions which may have found the fissile material. The way in which this is enacted is slightly different depending on where a potential solution is found. If it is found during the gradient descent portion then the individual is stored and a new individual is randomly selected to take its place, whereas the genetic algorithm is slightly different.

Let us assume we have n individuals being used in the genetic algorithm and, abiding by the method set out above, the individual with the best fit is chosen to be part of the population of solutions. As a result of this it is removed and replaced with a randomly selected individual, I. It is reasonable to assume that this new individual will have a higher cost functional associated with it than any other individual i.e. it

fits the data the least. Therefore it will have the lowest probability of being chosen during the crossover stage of the genetic algorithm.

Let I_k be individual with rank $r_k = n$ and let X represent the individual chosen at a certain crossover point. Going back to the chapter on the genetic algorithm the fitness of an individual is

$$S_i^{'} = \frac{n - r_i + 1}{n} \tag{5.3}$$

which for individual I_k is

$$S_k^{'} = \frac{1}{n}. \tag{5.4}$$

The probability of being chosen is this divided by the sum of all fitness values

$$F = \sum_{i=1}^{n} S_i^{'} \tag{5.5}$$

which is known as the Roulette method. Note that the numerator of the fitness simply cycles through the numbers 1 to n, meaning

$$F = \frac{1}{2}(n+1). \tag{5.6}$$

Therefore the probability of I_k being chosen at a specific crossover point is

$$P(X = I_k) = \frac{2}{n(n+1)}. \tag{5.7}$$

With a population of 30 this equates to a probability of 2.15×10^{-3}. However there are n instances during the crossover stage where it could be chosen. So the probability of never being chosen is

$$P(X \neq I_n \; over \; n \; instances) = \left(1 - \frac{2}{n(n+1)}\right)^n. \tag{5.8}$$

If we again use a population of 30 this gives us a probability of 0.937 that the lowest ranked individual is never chosen, and so it is a very small chance it will be. Also even if it is chosen the offspring it makes probably won't have a good fitness either, so any information has a low chance of surviving a further generation. Ideally we would want to introduce a lasting effect on the population with this new individual, otherwise we may as well simply remove the previous individual without replacement.

So we approach from the other side. Let $m \in \mathbb{Z}$ be such that $1 \leq m \leq n$. Say we have replaced a number of individuals during the above stage, and replaced them with

new individuals with worse fitness than the current ones. Then m will be the rank of the individual with the best fitness entering the population i.e. we will have replaced $n-m+1$ individuals. If $m=n$ we end up with the above probabilities. Next we need to calculate the probability than none of these new individuals will be chosen during the next crossover stage

$$P(X \not\geqslant I_m \text{ over } n \text{ instances}) = \left(1 - \frac{2}{n(n+1)} \sum_{i=m}^{n} i\right)^n \tag{5.9}$$

which, when simplified, becomes

$$P(X \not\geqslant I_m \text{ over } n \text{ instances}) = \left(\frac{m(m-1)}{n(n+1)}\right)^n. \tag{5.10}$$

We can now use this to our advantage. Say we decide that we want there to be a prescribed probability P_c that the new individuals will be chosen at least once during the crossover stage. Then using the above gives us a quadratic, the solution of which gives us our value of m

$$m = \frac{2n+3-\sqrt{(2n+3)^2 - 4(n+1)(2+n\sqrt[n]{1-P_c})}}{2}. \tag{5.11}$$

This will provide us with a target number of individuals to replace given a target probability. If we wanted $P_c = 0.9$ for a population of 30 then using the above we get $m = 23.2$, which would be rounded down, and so in this instance 8 individuals are replaced.

5.2 Ensemble algorithm results

In order to illustrate how the method acts, the cost functional at various stages of the method have been plotted for the simple case of 5.1. Figure 5.8 shows the minimum and maximum cost functionals for the steepest descent algorithm in the first 20 iterations of the method. As seen with the steepest descent algorithm from the previous chapter there is fast convergence followed by more of an oscillatory motion which begins to level out. There is still a slight downwards trend in this case but continuing much longer won't result in drastically better results.

The steepest descent method works on 5 separate individuals for 20 iterations each. Restricting our attention to one application of the steepest descent algorithm, the

starting point is stored. The the density distribution is stored at every 4th iteration, resulting in 6 density distributions stored for each individual the steepest descent works on. Therefore 30 density distributions are fed into the genetic algorithm, which makes up the starting population. No probability is utilised when sampling density distributions for the purposes of the genetic algorithm, which works on them for 200 iterations as shown in figure 5.9. Only the minimum cost functional is chosen and it shows the genetic algorithm is, at this stage, reducing the cost functional at most of the iterations.

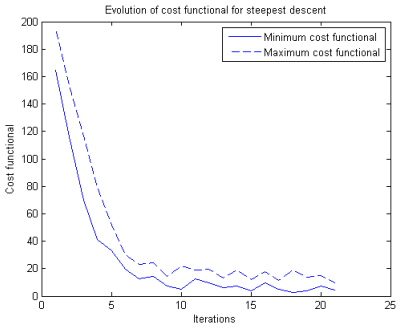

Figure 5.8: Cost functional for the steepest descent stage of the first cycle for 5.1.

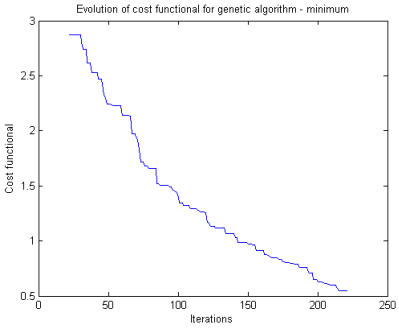

Figure 5.9: Minimum cost functional for the genetic algorithm stage of the first cycle for 5.1.

For this example the algorithm had to be run for 62 cycles before finding the prescribed number of solutions, the term cycle being used to describe enacting both stages of the steepest descent method and genetic algorithm. Figure 5.10 shows the minimum value of the cost functional over the 20 iterations of the steepest descent method in the final cycle. At this stage it is not decreasing magnitude, merely oscillating. However it is within the feasible domain for solutions as at iteration 13430 the best individual is replaced. This is why in figure 5.11, which is the maximum cost functional for the steepest descent method, only reaches iteration 13429, as after that there would be a large spike in magnitude. In figure 5.11 apart from a small spike at iteration 13424 thee cost functional is decreasing so the steepest descent method still manages to add to the variety.

Figure 5.12 shows the minimum cost functional for the genetic algorithm in the final cycle and there are several spikes indicating occasions of solutions being found throughout the 200 iterations. In figure 5.13 the minimum cost functional before the first spike has been occurred and it seems to decrease several times during this stage.

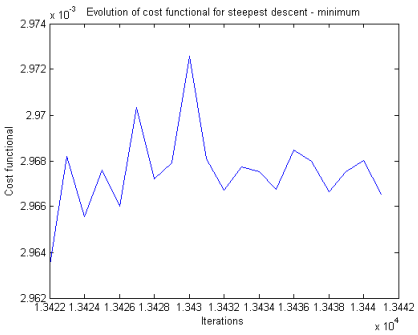

Figure 5.10: Minimum cost functional for the steepest descent stage of the final cycle for 5.1.

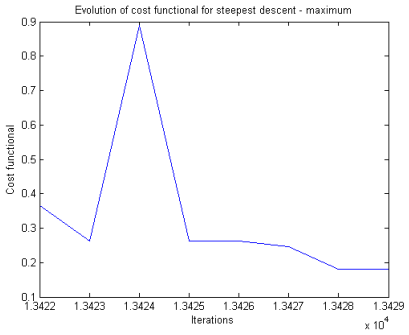

Figure 5.11: Maximum cost functional for the steepest descent stage of part of the final cycle for 5.1.

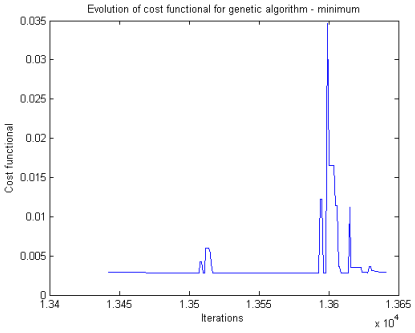

Figure 5.12: Minimum cost functional for the genetic algorithm stage of the final cycle for 5.1.

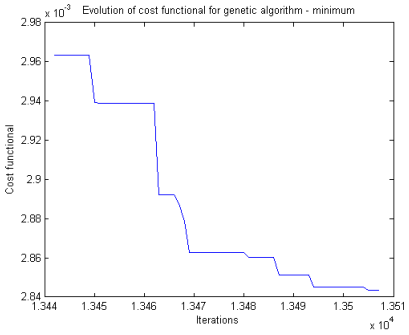

Figure 5.13: Minimum cost functional for the genetic algorithm stage of part of the final cycle for 5.1.

After running the above method we will end up with a population of solutions and so we need to settle on the best way to analyse them. In other scanning techniques, such as X-ray, there will be an operator who is trained to identify any abnormalities based on whatever it is they are looking for. Since we have a number of solutions it would be time consuming for someone to look at each one so it would be preferable to find a way to quantify the results.

Another possible problem is that we expect to find small amounts of lead in the reconstructions sometimes - a larger object could be realised as a small piece of lead without changing the data that much, so the mere presence of lead in the reconstruction won't be indicative of lead actually being present. Simply measuring the total volume of lead is not a viable option due to many possible smaller pieces being found.

Instead we turn to a form of lens. Once we have a reconstruction we look for objects of a certain size, say a cube of length $p$cm on each side, and place this object at every possible point it could exist in the cargo container. Then we measure the

average density of an object placed at each point and focus on the maximum achieved. In such a way we can attempt to figure out how likely it is that a lead object of that size is present in the cargo container. Given that lead objects are likely to turn up for any example we must also use the data created from the same example except without lead being present.

One way to think of the the lens is as follows. Define a domain L to be a cube that is at least the size of one voxel and cannot be any fraction of a voxel - so in voxel measurements it would be $1 \times 1 \times 1$ or $2 \times 2 \times 2$ and so on. Now place it somewhere in the cargo container domain so that it lines up with the voxel arrangement - any voxel within L will not have any part sticking out. Each voxel within L will have a corresponding density component. Define $\boldsymbol{\rho}_L$ to be the density components residing within domain L. Next take the average of the components of $\boldsymbol{\rho}_L$ and record it - this is the average density within L, but it is only at one location within the cargo container. So L is placed at every possible location within the cargo container and the average density within recorded. Then the maximum is taken.

The lens is an attempt at finding out the maximum density achieved for a small object.

Over the course of the next few sections there are various examples analysed. It starts with a simple case and gradually adds more objects to increase the complexity of the cargo container. One possible realistic example is looked into but it is still an approximation to reality. The restriction to distinct domains will not always hold - cargo containers will be varied and many may contain objects that resemble smoother domains. However there will be many cargo containers with some order to them - the use of stacked pallets will create large objects of roughly constant density, with distinct differences between them and their surroundings. So while the the most complex example analysed here may represent a real-life cargo container, the examples will not be representative of a general cargo container.

5.2.1 A simple example

We begin by looking at the case shown in figure 5.1, which is that of a lead cube of volume $25 \times 25 \times 25\text{cm}^3$ with only one wooden object and one metal object, all placed at random locations around the cargo container. Using a relatively accurate weighing

system would provide us with an observed weight (or mass) of 7.44×10^7g which, when fed into the relation calculated from figure 5.7, gives us a target cost functional of $\mathcal{J}_{tol} = 3.472 \times 10^{-3}$. When the actual good solutions were calculated the related cost functionals were plotted in figure 5.4 where the actual target cost functional should be larger - slightly over 5×10^{-3}. Since the calculated tolerance is smaller this means some good solutions will be missed, unless the algorithm is forced to adapt due to it due to the tolerance being unlikely to be reached. However in figure 5.14 we see that the calculated tolerance remains untouched as all individuals have a cost functional of 3.472×10^{-3} or less and a population of solutions has been successfully calculated, albeit covering a smaller range of cost functionals than all the correct solutions.

Using the same method on the density distribution without lead reveals that the usual good solutions reach a maximum cost functional of approximately 1.7×10^{-3}, but the calculated cost functional is $\mathcal{J}_{tol} = 3.08 \times 10^{-3}$, based on the weight, and when this is used the cost functionals for the resultant individuals (figure 5.15) fail to reach the intended range of good solutions. Therefore in this case it is possible that none of the solutions will resemble the actual density distribution.

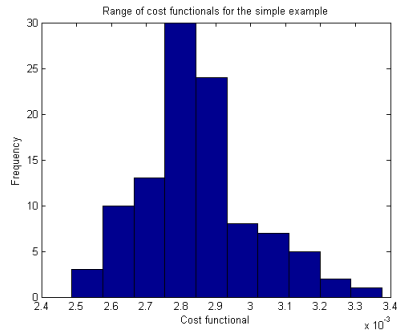

Figure 5.14: Histogram of the cost functionals when algorithm run for 5.1.

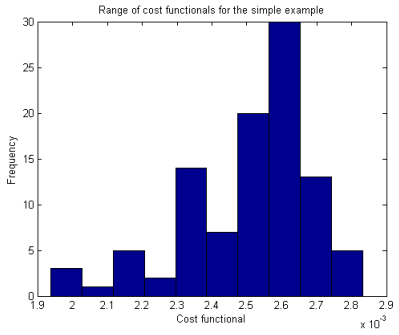

Figure 5.15: Histogram of the cost functionals when algorithm run for 5.1 without lead.

The results involving the lens are shown in figures 5.16 to 5.23, with those on the left relating to lead being present in the actual density distribution and those on the right having it missing. We start by looking for smaller objects but since the voxel size is 10cm we don't look for objects of that size - it is very likely there will be individual voxels somewhere in the cargo container with a high density.

Beginning with those on the left we see that almost all solutions contain and object with density very close to lead and size 20cm on each edge. On its own this would

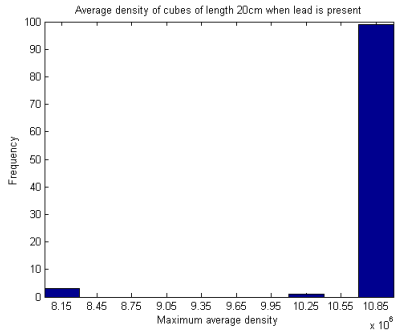

Figure 5.16: Histogram of the average density of cubes of size 20cm when using the data from 5.1.

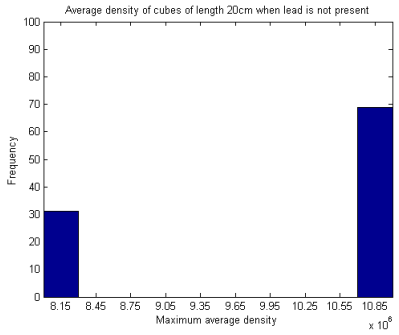

Figure 5.17: Histogram of the average density of cubes of size 20cm when using the data from 5.1 without lead.

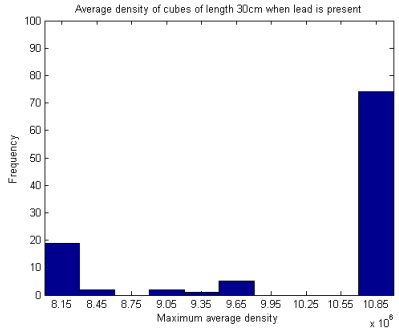

Figure 5.18: Histogram of the average density of cubes of size 30cm when using the data from 5.1.

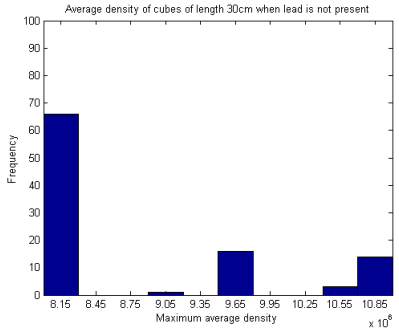

Figure 5.19: Histogram of the average density of cubes of size 30cm when using the data from 5.1 without lead.

heavily imply that an object of size 20cm is present. Moving onto figure 5.18 we find that most solutions also contain an object of size 30cm on each edge which reaches a density close to lead. However a small number only reach a density of metal. This may imply less strongly that an object of size 30cm is present. What is interesting is that there is a similar level of objects found that are 40cm in size in figure 5.20. Since this is much larger than the actual size of 25cm it may be that such a level is not indicative of an object being present. Finally as we move onto objects of 50cm we see that the highest density achieved is that of some metals, thus we would conclude an object of such size is not present.

The figures which depict the case without the lead being present (on the right) show the same trend except sooner. As soon as we look for an object of size 30cm or higher the majority of solutions favour a metallic one being the most likely, which is promising. Figure 5.17 is the important one for comparison purposes as around 70% of solutions contain an object of size 20cm being a density close to lead.

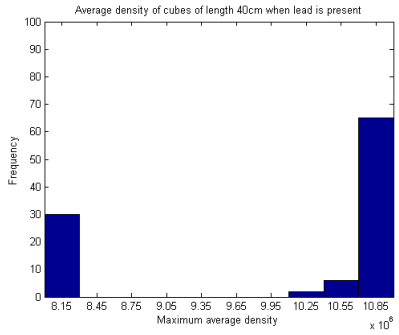

Figure 5.20: Histogram of the average density of cubes of size 40cm when using the data from 5.1.

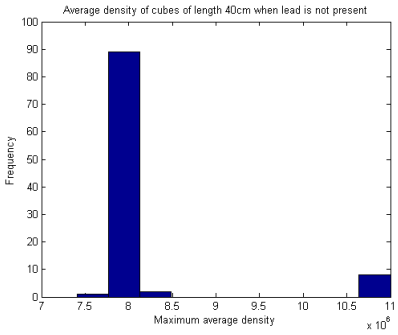

Figure 5.21: Histogram of the average density of cubes of size 40cm when using the data from 5.1 without lead.

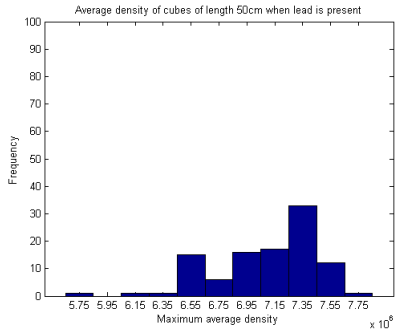

Figure 5.22: Histogram of the average density of cubes of size 50cm when using the data from 5.1.

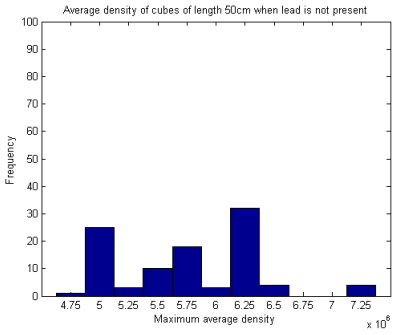

Figure 5.23: Histogram of the average density of cubes of size 50cm when using the data from 5.1 without lead.

Although this case is a simple (and most likely unrealistic) scenario it may provide us with insight as to what to expect in later cases. From this case alone we could be inclined to say that 70% of solutions indicating a high density object is not a large enough proportion to indicate anything is there, we may require a higher proportion of say larger than 90% to say with more authority on the density distribution. Or it could be that it is not possible to identify such high density material. We need to look at more cases in order to decide which conclusion is more likely.

Another simple example is shown in figure 5.24, where again there is a small piece of lead only this time it is 30cm on each side. There is also one piece of wood and one piece of metal elsewhere in the domain.

When the data was compared to good solutions the tolerances of 2.8×10^{-3} and 2.3×10^{-3} were found to be good values for the case with and without lead respectively. The weight of this cargo container is 7.45×10^{7}g which, when fed into the relationship

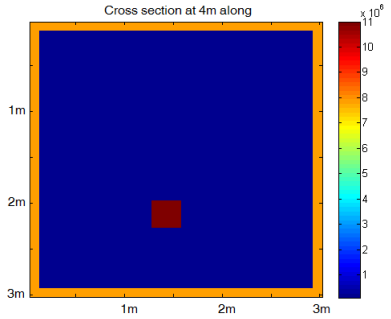

Figure 5.24: Cross section of the cargo container containing the small lead object as another uncluttered example.

calculated, returned tolerances of $\mathcal{J}_{tol} = 3.76 \times 10^{-3}$ and $\mathcal{J}_{tol} = 3.08 \times 10^{-3}$ respectively. Since the calculated tolerance has overestimated what we would call the correct tolerance the assumption would be that the method would have no problem finding solutions beneath the prescribed tolerance.

However in figures 5.25 and 5.26 we see that the ranges of cost functionals all lie above the calculated tolerances. This means that the algorithm has found it difficult to reach the prescribed tolerance and therefore has worked, over several cycles, to increase it. This could be due to the radial basis mesh being too coarse to be able to correctly model the actual distribution but based on the size and shapes of the objects within the container and the mesh chosen this shouldn't be the case. Another option is that the algorithm is working slower than expected and simply requires more time to reach the prescribed tolerance before adjusting it. It means that the solutions acquired in this case do not lie within the range of cost functionals for good solutions.

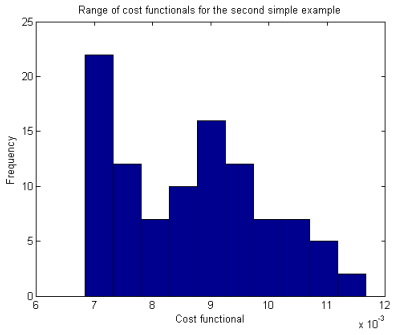

Figure 5.25: Histogram of the cost functionals when algorithm run for 5.24.

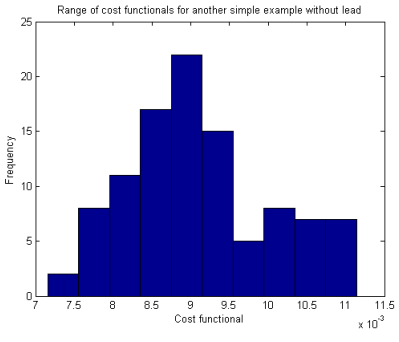

Figure 5.26: Histogram of the cost functionals when algorithm run for 5.24 without lead.

The histograms 5.27 to 5.34 show the results in a quantitative manner - those

on the left are for when a piece of lead is present and those on the right are the same distributions except without lead. Beginning with figure 5.27 it shows that all individuals contain at least one cube of size 20cm on each side within their domain. Given that the actual object is 30cm on each side this is to be expected. There is a slight drop off when using a lens of that size though, as shown in figure 5.29. Over 90% of individuals have found an object of that size of density close to lead, which is allowable, but there is a similar level when using a 40cm sized lens (figure 5.31). In the previous example when the lens used was larger than the actual lead object it resulted in a decrease to around 70% success rate whereas it lead to barely any noticeable difference, which is most likely due to the placement of the lead. In figure 5.1 it is closer to the edge (and the sensors) than in this case and so we are more likely to achieve good results. Here it is buried more towards the middle of the cargo container and so the recovered size might be less accurate.

Moving onto figure 5.33 the number of individuals that contain an object relatively close to the density of lead of size 50cm on each side is approximately 40%. Of course this depends on what we define 'relatively close', here it is any density over 10g/cm^3 but if we were to relax this to be close to 9.5g/cm^3 then more than half of the population correctly model the density distribution. As we increase the size of the lens this trend continues in the same way.

For the cases without lead being present in the container we start with figure 5.28. When using a lens of size 20cm over 90% of individuals manage to recover an object of density close to lead (which is also higher than when using the same sized lens in the previous simple example). When the lens size is increased to 30cm the percentage drops to around 80 - still relatively high. Similarly when using a lens of size 40cm on each side the percentage remains at around 70. It is only in figure 5.34 when the percentage has dropped in any significant manner, and that is the case where we look for objects of size 50cm on each side.

Overall the figures on the right follow the same trend as those on the left, with only a slight noticeable difference between the two. The aim of this section is to home in on a lens size which could be used to minimise false positives and negatives. In the previous example a lens size of $30 - 40$cm on each side would be useable if we were to base our results on what the majority of the population agree is the most likely

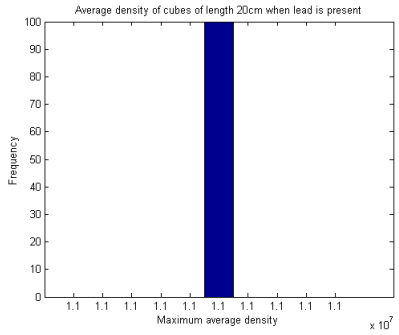

Figure 5.27: Histogram of the average density of cubes of size 20cm when using the data from 5.24.

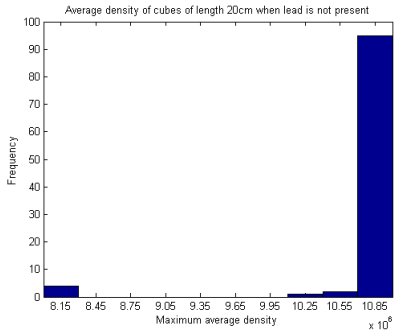

Figure 5.28: Histogram of the average density of cubes of size 20cm when using the data from 5.24 without lead.

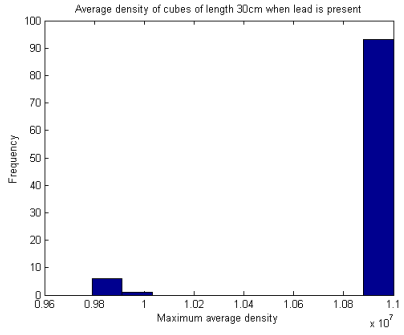

Figure 5.29: Histogram of the average density of cubes of size 30cm when using the data from 5.24.

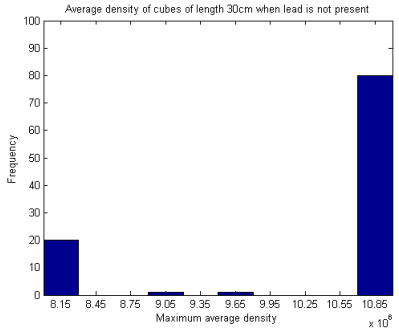

Figure 5.30: Histogram of the average density of cubes of size 30cm when using the data from 5.24 without lead.

maximum density. Using either lens size results in most of the population returning a maximum density close to lead if it is present in the cargo container, whereas if there is no lead present most individuals agree that the maximum density is around 8g/cm^3, which is defined to be domain D_3, relating to the metal blocks made of iron or steel. However neither lens size would be suitable for this second simple example as there isn't as much of a noticeable difference.

The first simple example is unrealistic as the lead box (and therefore fissile material) has been placed very close to the edge with no objects nearby which could shield it from assumed scans if we assume it is being smuggled in. However the second example doesn't contain any objects to shield it either, simply relying on distance from the edges to remain undetected, so neither might be practical. Alternatively it is that failure of the algorithm to reach the region of feasible solutions that has lead to unreliable data. It is possible that a lens size of 50cm could work here but we would have to decide on what density is relatively close to lead and how much of the population has to agree

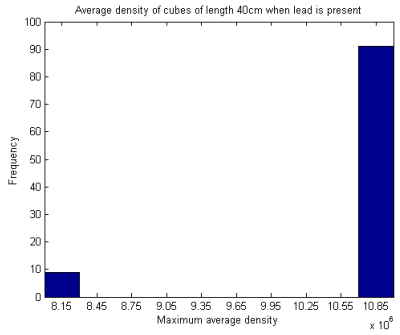

Figure 5.31: Histogram of the average density of cubes of size 40cm when using the data from 5.24.

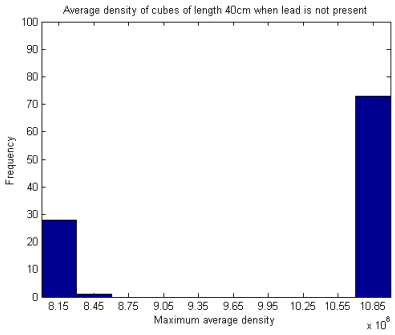

Figure 5.32: Histogram of the average density of cubes of size 40cm when using the data from 5.24 without lead.

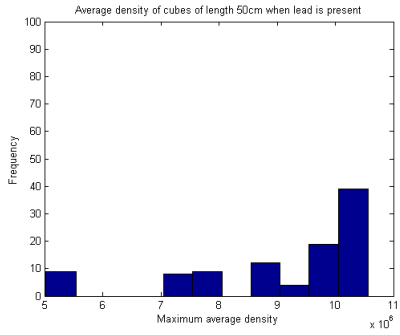

Figure 5.33: Histogram of the average density of cubes of size 50cm when using the data from 5.24.

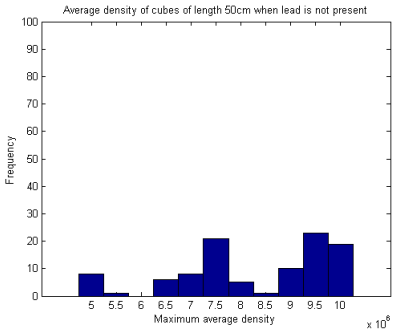

Figure 5.34: Histogram of the average density of cubes of size 50cm when using the data from 5.24 without lead.

to make it correct. These are decisions to be made based on the remaining parts of this section

5.2.2 A semi-cluttered example

Below in figure 5.35 is a slightly more complex example than those in the previous section. A few more objects of wood and metal have been added to the domain to make it slightly more cluttered and that is reflected in the increase in weight to 1.03×10^8g. Using this the calculated tolerance for the cost functional is $\mathcal{J}_{tol} = 6.8 \times 10^{-2}$, although when using a variety of correct solutions it seems that the tolerance should be set to approximately 9×10^{-2}. There are correct solutions which have the cost functional of around 6×10^{-2} and below but the algorithm finds it difficult to reach them - leading to figure 5.36 in which the tolerance has been increased to approximately 1.4×10^{-1} and the minimum cost functional found is around 1.2×10^{-1}, meaning that we haven't managed to reach the region of good solutions for this case.

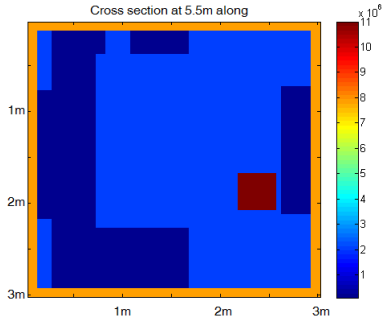

Figure 5.35: Cross section of the cargo container containing the small lead object as a semi-cluttered example.

When there is no lead in the cargo container the calculated tolerance is found to be $\mathcal{J}_{tol} = 6.7 \times 10^{-2}$. However the actual good solutions inhabit a region with an approximate maximum cost functional of 5×10^{-2}, which is much less than the cost functional for the case with lead. Since the lead object is rather close to the sensors it has a large impact on the data collected from the sensors, and this could lead to better results. Also since the calculated tolerance is above what it should be, the algorithm should be able to find solutions below it. Unfortunately it must take too long to do so as the tolerance must be increased at various times in the algorithm according to figure 5.37 where the tolerance has been increased to 9.5×10^{-2}, and it fails to reach the region of good solutions. This has happened both here and in the previous example and since in both instances the algorithm is able to eventually reduce the cost functional to a certain degree the main fault must lie with the alteration of the calculated cost functional - there is not enough time spent trying to reach the tolerance before increasing it.

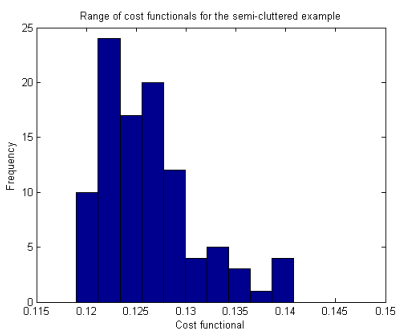

Figure 5.36: Histogram of the cost functionals when algorithm run for 5.35.

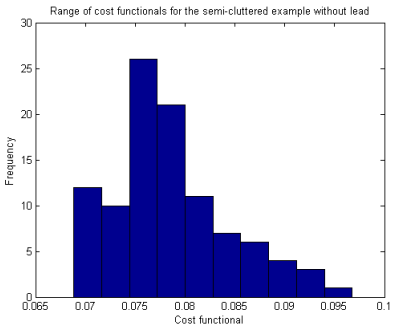

Figure 5.37: Histogram of the cost functionals when algorithm run for 5.35 without lead.

The results can be found in figures 5.38 to 5.47. Beginning with figure 5.38 we see that when using a lens of size 40cm on each side there is at least one object in each individual that is the density of lead. Given that the actual object is of size 30cm on each side the algorithm has overestimated the size of the object, like it has done on previous occasions. Figure 5.38 shows an identical result except where the data used did not include the lead object, meaning we have a false positive - at least lead one object has been reconstructed in the domain without lead ever being present. Such an occurrence is expected after increasing the number of objects in the domain as we have done.

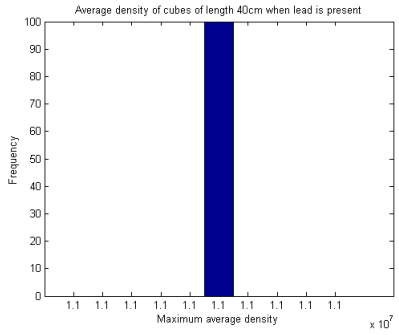

Figure 5.38: Histogram of the average density of cubes of size 40cm when using the data from 5.35.

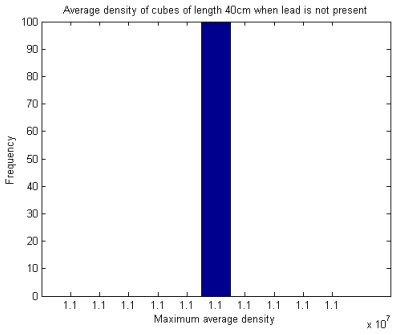

Figure 5.39: Histogram of the average density of cubes of size 40cm when using the data from 5.35 without lead.

Moving onto the pair of figures 5.40 and 5.41, which look at using a lens of size 50cm on each side with the former containing lead and the latter not, a difference has begun to emerge. When lead is present all individuals manage to contain a lead object in the reconstruction, when the lead is taken away not all individuals continue to identify the object as completely lead, some have a slightly decreased density. However the decrease is very slight, equating to 1g/cm^3 change. To say this is a significant change may lead to very stringent rules on identifying lead.

This might not give us all we need and so instead we look at figures 5.42 and 5.43. In this case almost all individuals have correctly identified the lead, even though it would be 8 times the volume of the actual lead object. In the case of the data without lead it seems the reconstruction has reduced enough to be firmly in a different domain range, that of iron and steel (around 8g/cm^3). This should constitute a significant enough decrease to be able to distinguish between the two possibilities.

As we look for larger objects for the distribution without lead (figures 5.45 and

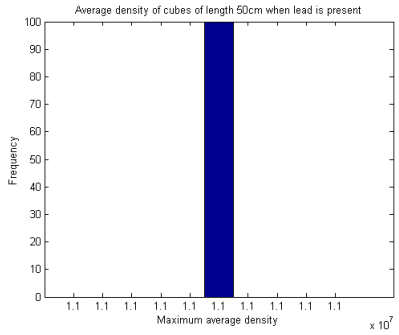

Figure 5.40: Histogram of the average density of cubes of size 50cm when using the data from 5.35.

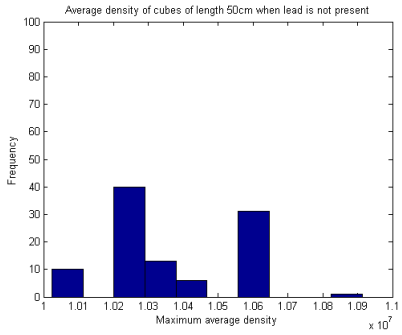

Figure 5.41: Histogram of the average density of cubes of size 50cm when using the data from 5.35 without lead.

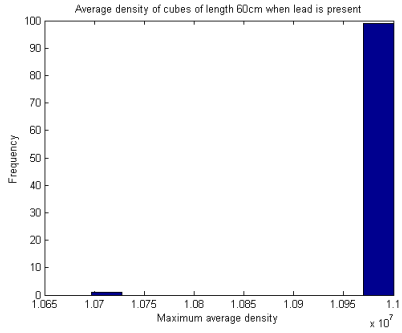

Figure 5.42: Histogram of the average density of cubes of size 60cm when using the data from 5.35.

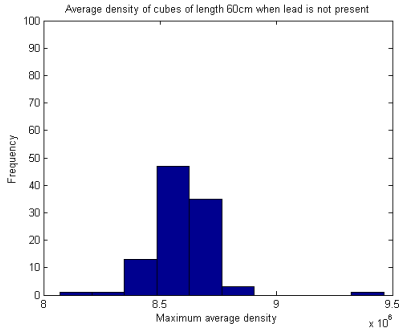

Figure 5.43: Histogram of the average density of cubes of size 60cm when using the data from 5.35 without lead.

5.47) the range of densities continues on a downward trend as expected. What is more interesting are figures 5.44 and 5.46. When we look for objects of size 70cm on each side the results seem to resemble those in figure 5.41 - a slight decrease but centering around a density of 10g/cm^3. It is only when a lens of 80cm is used that there is a significant decrease to a lower density domain.

Overall this case involves an object of size 30cm being present close to the sensors in a semi-cluttered domain. When searched for the algorithm identifies objects up to size 70cm on each side that could easily resemble lead. Without any lead being present there are false positives involving objects up to size 50cm on each side, thus providing us with a region of identification for the lens size (60-70cm) for this case which should avoid both false positives and negatives.

The reason behind this could be due to the placement of the lead - inside a large wooden block in figure 5.35. This block is also close to the sensors and it is likely to be reconstructed as a higher density object leading to false positives. Other objects

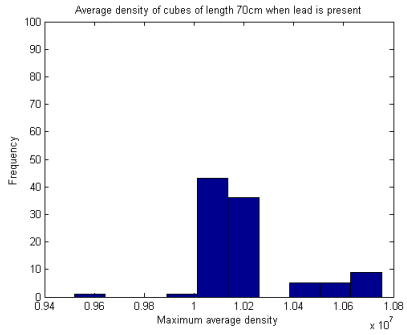

Figure 5.44: Histogram of the average density of cubes of size 70cm when using the data from 5.35.

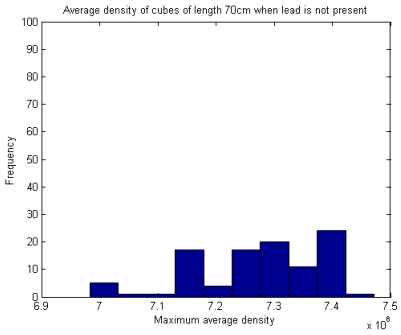

Figure 5.45: Histogram of the average density of cubes of size 70cm when using the data from 5.35 without lead.

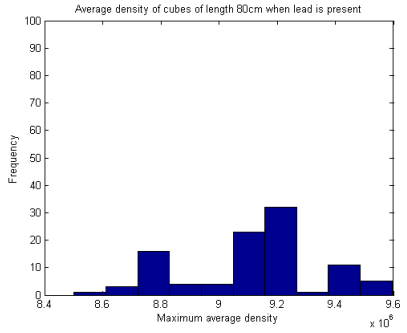

Figure 5.46: Histogram of the average density of cubes of size 80cm when using the data from 5.35.

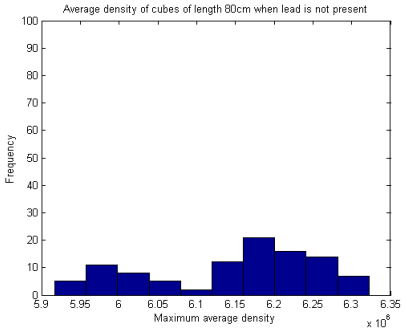

Figure 5.47: Histogram of the average density of cubes of size 80cm when using the data from 5.35 without lead.

may also follow this trend, but the addition of an actual lead object may cause the reconstructed lead to increase in size. This is the first example of how an increase in the number of objects (and also the cargo container weight) leads to an increase in the size of the lead object recovered.

Another semi-cluttered example can be seen in figure 5.48 with a weight of 9.96×10^7g, which leads to a computed tolerance for the cost functional of $\mathcal{J}_{tol} = 6.2 \times 10^{-2}$. However the actual tolerance should be around 3.3×10^{-2}. Similarly with the case without the lead present the computed tolerance is $\mathcal{J}_{tol} = 6.1 \times 10^{-2}$ whereas the actual tolerance should be approximately 3.2×10^{-2}. Both have been prescribed tolerance levels that are higher than they should be and both, as shown in figures 5.49 and 5.50, require an increase to the cost functional in order to work in a shorter time frame, neither reaching the region of cost functionals associated with correct results. Therefore this is another example where the solutions achieved may not be the best, whether due to not allowing the algorithm to run for long enough before increasing the

tolerance, or because the radial basis mesh is to coarse to allow lower cost functionals. In either case the results it generates are in figures 5.51 to 5.58.

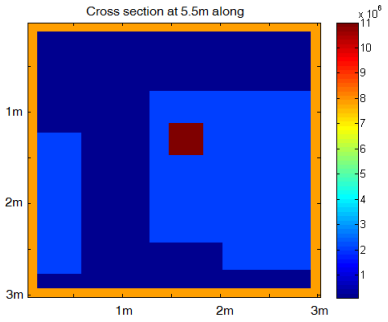

Figure 5.48: Cross section of the cargo container containing the small lead object as another semi-cluttered example.

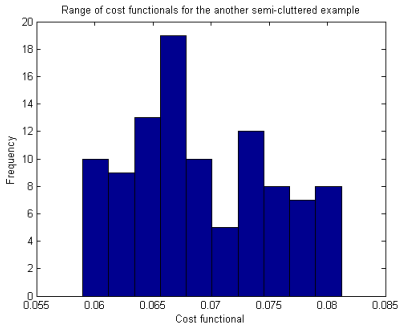

Figure 5.49: Histogram of the cost functionals when algorithm run for 5.48.

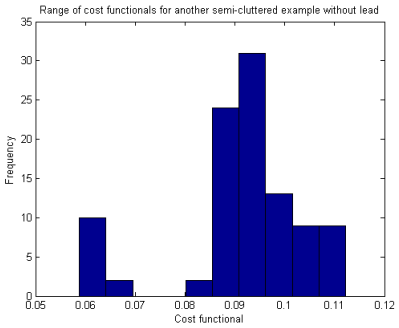

Figure 5.50: Histogram of the cost functionals when algorithm run for 5.48 without lead.

Here the same number of wooden and metallic boxes as the previous semi-cluttered example have been selected, except their size and locations are randomly chosen which leads to a slightly lower weight than for the previous example. For the moment let us restrict ourselves to the cases where lead is present in the actual density distribution. As with the previous case when we use a lens of size 40cm on each side all individuals have at least one object made of lead (figure 5.51), but increasing to a lens of size 50cm (figure 5.53) reveals a slight difference similar to the one seen before in figure 5.42. As in that case the slight reduction in density is not significant enough to be considered a negative result, although such a change occurs at a smaller size here.

Figure 5.55 shows the data when using a lens of size 60cm and it is here where there could be some significant change. The lower densities recovered are close to 9g/cm^3, which is encroaching upon the lower density domain of iron and steel. However the

upper bound remains relatively close to lead. While we may have still achieved a consensus from the population at least some individuals continue to identify a lead object of size 60cm on each side present. It is only when increasing to a lens of size 70cm on each side (figure 5.57) that we reach a negative result - all individuals have a maximum average density of around 9.5g/cm^3 and around 70% reach below 9g/cm^3.

Contrast this to the case without lead and we find a slightly different result. Using a lens of 40cm (figure 5.52) still returns all individuals as having identified lead, but increasing the size of the lens to 50cm (figure 5.54) results in a shift towards the lower densities. As seen in figure 5.55 the lower densities may be encroaching on the lower domain but the upper boundary remains close to lead, and so this is still a positive result for lead. It is only when the lens size is increased to 60cm (figure 5.56) that a negative result could appear. All individuals return a maximum density of less than 10g/cm^3 with a significant proportion returning 8g/cm^3. Figure 5.58 goes further with a lens size of 70cm and it returns a definite negative result as all maximum densities are less than 8g/cm^3.

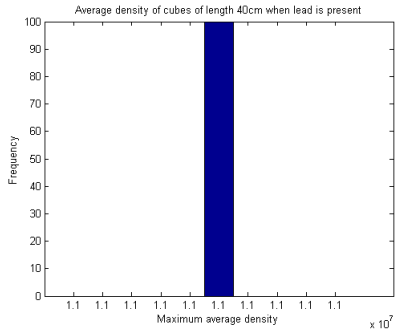

Figure 5.51: Histogram of the average density of cubes of size 40cm when using the data from 5.48.

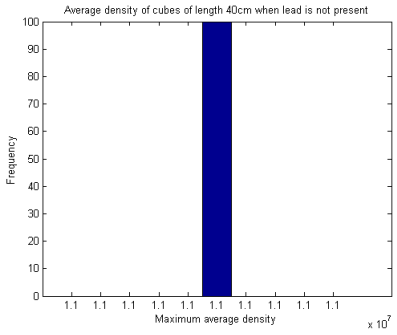

Figure 5.52: Histogram of the average density of cubes of size 40cm when using the data from 5.48 without lead.

For this case it seems that using a lens of size 60cm is key, as it could differentiate between a cargo container with lead and one without. Any smaller lens leads to false positives and anything larger returns false negatives. Having a single sized lens as an indicator ties into the first two simple examples as there is a very thin range for false positives and negatives, whereas the previous semi-cluttered example had a larger range of 60-70cm. This could be due to the larger weight of the cargo container in that case, or because the lead object was closer to a sensor. In either case we are on track to identifying a trend for the indicator for high density material in the cargo

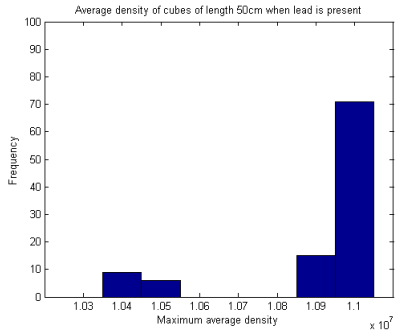

Figure 5.53: Histogram of the average density of cubes of size 50cm when using the data from 5.48.

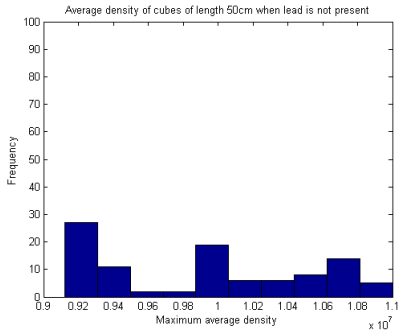

Figure 5.54: Histogram of the average density of cubes of size 50cm when using the data from 5.48 without lead.

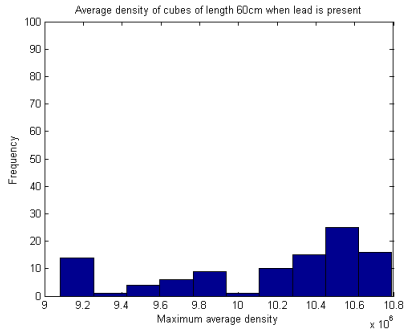

Figure 5.55: Histogram of the average density of cubes of size 60cm when using the data from 5.48.

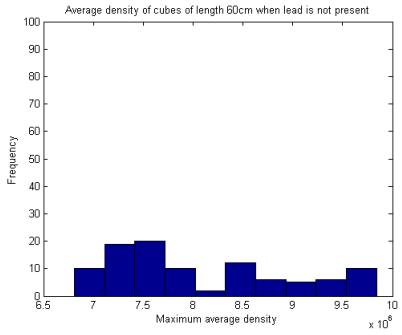

Figure 5.56: Histogram of the average density of cubes of size 60cm when using the data from 5.48 without lead.

container.

5.2.3 A cluttered example

Next we look at the more cluttered example first shown in figure 5.2. Here we have increased the number of wooden and metal blocks randomly scattered around the domain and the weight reflects that - at 1.10×10^8g it is the heaviest cargo container so far. Using this the calculated cost functional tolerance is found to be $\mathcal{J}_{tol} = 8.58 \times 10^{-2}$ and, according to figure 5.59, the algorithm seems to stick with it. When restricting ourselves to good solutions they seem to range up to around 1×10^{-1}, but still several solutions reach lower than 8.58×10^{-2}. So in this case the algorithm has managed to find results in the valid range of cost functionals. The same can't be said for the case without lead as this has a similar calculated tolerance of $\mathcal{J}_{tol} = 8.48 \times 10^{-2}$ but the algorithm has increased this at various stages, as shown in figure 5.60. Also several good solutions only reach up to the cost functional of 1×10^{-1} whereas all individuals

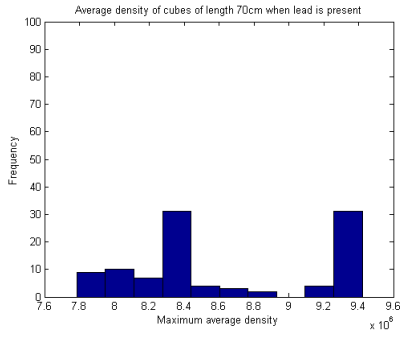

Figure 5.57: Histogram of the average density of cubes of size 70cm when using the data from 5.48.

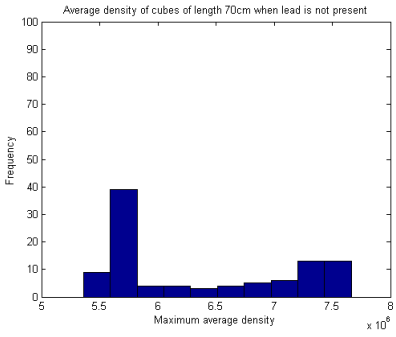

Figure 5.58: Histogram of the average density of cubes of size 70cm when using the data from 5.48 without lead.

found from the algorithm have larger cost functionals.

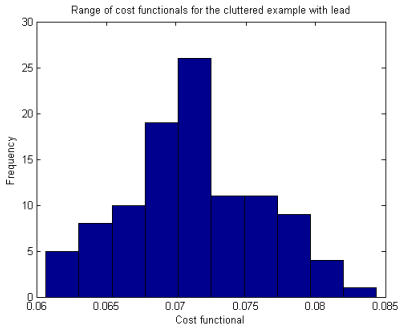

Figure 5.59: Histogram of the cost functionals when algorithm run for 5.2.

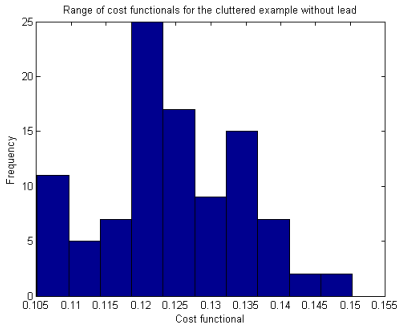

Figure 5.60: Histogram of the cost functionals when algorithm run for 5.2 without lead.

The first results are seen in figures 5.61 and 5.62, in which a lens of 40cm has been used and has gained us a positive result whether lead is present in the cargo container or not. Instead, if we use a lens of size 50cm we start to see a slight distinction between the results in figures 5.63 and 5.64. However since there is a very limited range in figure 5.64 and it remains near to the density of lead it will still be identified as a false positive.

It is only when moving onto a lens of size 60cm when we can see a proper distinction between the two. The densities in figure 5.65 have managed to remain close to lead whereas those in figure 5.66 have moved much closer to the density of domain D_3, even though a few individuals may achieve something close to a positive result.

We can say with relative confidence that a lens of size 70cm (figures 5.67 and 5.68) will provide a negative result in both instances, and so a lens of size 60cm will be the key to distinguishing between false positives and negatives in this case. This is

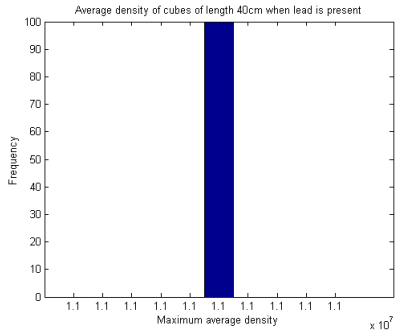

Figure 5.61: Histogram of the average density of cubes of size 40cm when using the data from 5.2.

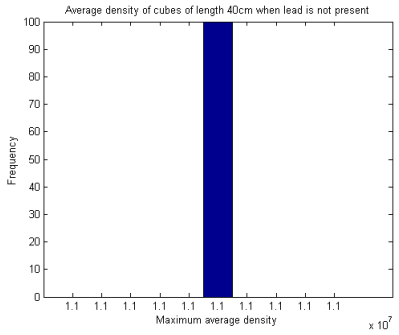

Figure 5.62: Histogram of the average density of cubes of size 40cm when using the data from 5.2 without lead.

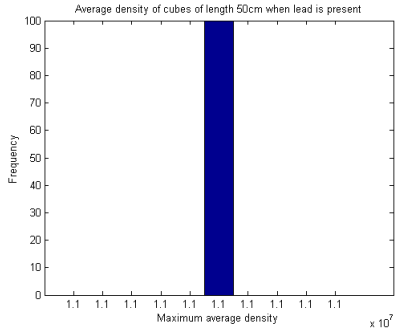

Figure 5.63: Histogram of the average density of cubes of size 50cm when using the data from 5.2.

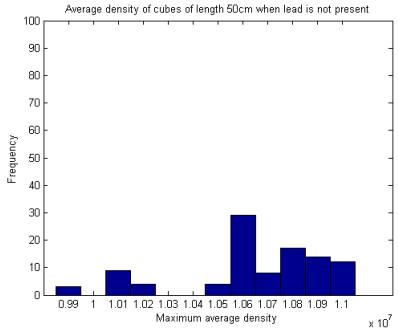

Figure 5.64: Histogram of the average density of cubes of size 50cm when using the data from 5.2 without lead.

interesting as the previous record for the highest weight (the first semi-cluttered example) had a slightly larger range in which we could distinguish between false positives and negatives, lending credence to the theory that it is the size and location of the lead object which makes the difference, although this will be unknown beforehand. Fortunately it seems that a lens of size 60cm works for the most recent examples.

The second cluttered example is shown in figure 5.3 with a weight of 1.03×10^8g in which we have increased the number of both wooden and metal blocks but, due to the random size and locations, we end up with a similar weight to that of the first semi-cluttered example, although in this example there are more metal blocks near to the lead and so will most likely provide slightly different results. In this case the calculated tolerance is 6.96×10^{-2} whereas, according to the good results compared to the data, it should be approximately 4.1×10^{-2}. In addition to that the algorithm has not managed to stick with the calculated tolerance, and has increased it to a degree that very few solutions have a cost functional reaching it (figure 5.69). The same

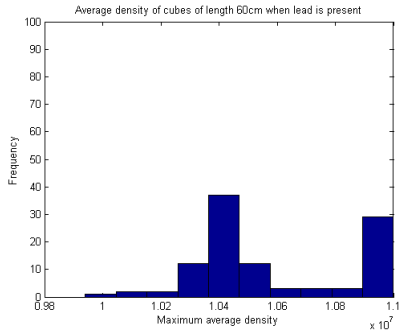

Figure 5.65: Histogram of the average density of cubes of size 60cm when using the data from 5.2.

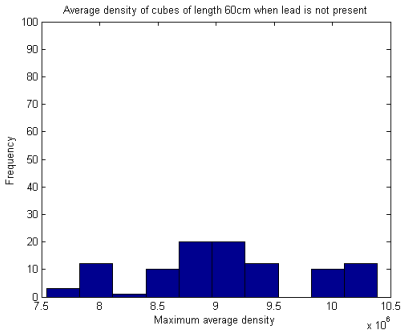

Figure 5.66: Histogram of the average density of cubes of size 60cm when using the data from 5.2 without lead.

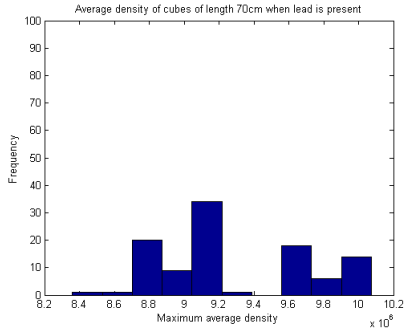

Figure 5.67: Histogram of the average density of cubes of size 70cm when using the data from 5.2.

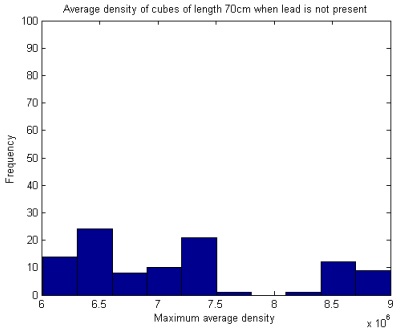

Figure 5.68: Histogram of the average density of cubes of size 70cm when using the data from 5.2 without lead.

thing has happened to the case without the lead being present (figure 5.70) where the tolerance should be around 4×10^{-2} but has been calculated to be $\mathcal{J}_{tol} = 6.92 \times 10^{-2}$ and the algorithm has increased it to around 1.1×10^{-1}. So in neither case should we expect the best solutions.

Beginning with figures 5.71 and 5.72 we see that, with or without the presence of lead, using a lens of size 40cm will net us a positive result, which agrees with what we have seen before in the more recent examples. Similarly an increase of the lens to 50cm does not provide a high distinction between the two results as in both figures 5.73 and 5.74 almost all individuals lie firmly close to the density of lead. In fact figure 5.73 contains a few individuals with lower densities than those in its counterpart, but due to the low frequency of said individuals it will be assumed that we can disregard such outliers.

In previous examples it is by using a lens of size 60cm which provided us with a distinction between false positives and negatives. Unfortunately it is not so obvious

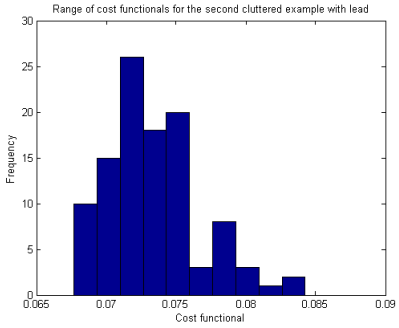

Figure 5.69: Histogram of the cost functionals when algorithm run for 5.3.

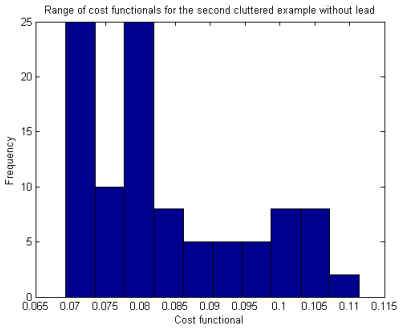

Figure 5.70: Histogram of the cost functionals when algorithm run for 5.3 without lead.

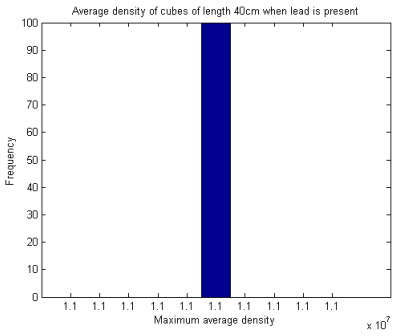

Figure 5.71: Histogram of the average density of cubes of size 40cm when using the data from 5.3.

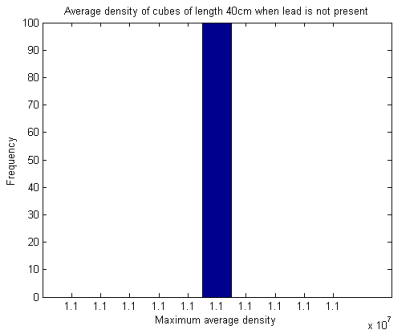

Figure 5.72: Histogram of the average density of cubes of size 40cm when using the data from 5.3 without lead.

here. Both figures 5.75 and 5.76 have densities over a large range, almost reaching lead on the upper boundary and the domain D_3 on the lower boundary, with the outliers mentioned above still present. The guidelines for distinction between false positives and negatives being developed throughout this section are rendered insufficient for the current result and so they must be refined.

An interesting aspect of the distribution of the two figures is the shape. Figure 5.75 is skewed more negatively whereas figure 5.76 has, if anything, a slight positive skew. This motivates us to find a percentage of individuals that reach above a certain density, which will be the aim of the final section of this chapter after one more example. Figures 5.77 and 5.78 have been included to show that a lens of 70cm will only lead to negative results in either case.

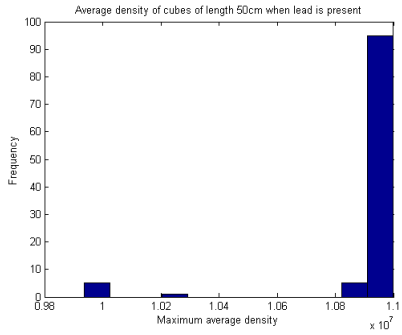

Figure 5.73: Histogram of the average density of cubes of size 50cm when using the data from 5.3.

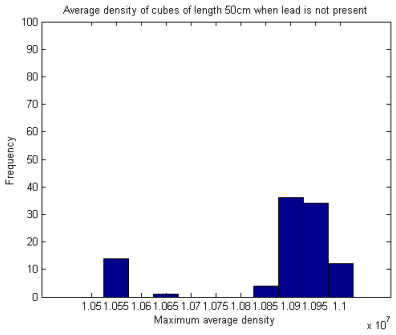

Figure 5.74: Histogram of the average density of cubes of size 50cm when using the data from 5.3 without lead.

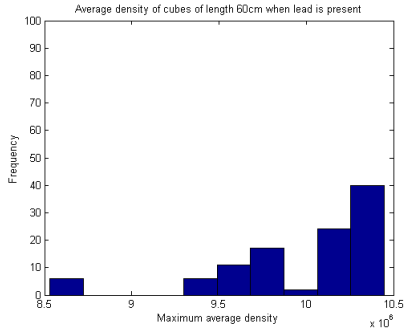

Figure 5.75: Histogram of the average density of cubes of size 60cm when using the data from 5.3.

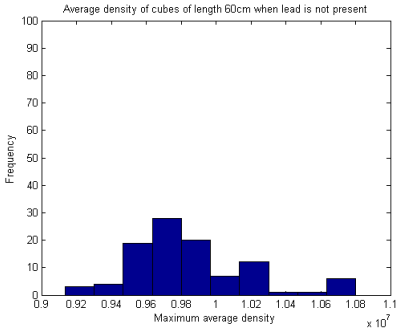

Figure 5.76: Histogram of the average density of cubes of size 60cm when using the data from 5.3 without lead.

5.2.4 A realistic example

Finally we attempt to approach a slightly more realistic example. So far we have looked at examples where the densities of each object has fit into the domains we have defined i.e. wood and metal, which would not happen in real life. Instead for this example then background of the cargo container has been populated with objects of varying size with a density in the range $(0, 5)\text{g/cm}^3$, resulting in an overall weight of $1.56 \times 10^8\text{g}$. Since the densities of each object does not fit into the domains defined in the reconstruction an exact solution is not possible. If it were possible then the solutions for the instances with and without lead would be 1.2×10^{-1} and 1×10^{-1} respectively. Extrapolating from the relationship developed earlier from figure 5.7 we instead get target cost functionals of approximately $\mathcal{J}_{tol} = 1.9 \times 10^{-1}$ for both. Using these in the algorithm (which increases them as required) result in a population with cost functionals shown in figures 5.80 and 5.81 respectively.

An increase in the tolerance should be expected due to inability to reach the actual

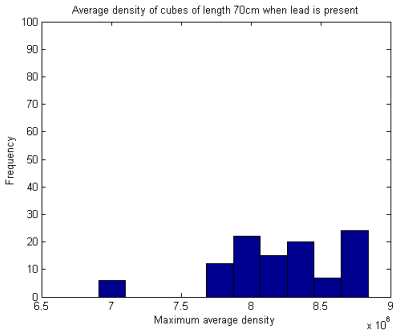

Figure 5.77: Histogram of the average density of cubes of size 70cm when using the data from 5.3.

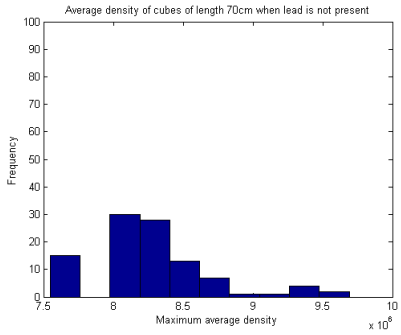

Figure 5.78: Histogram of the average density of cubes of size 70cm when using the data from 5.3 without lead.

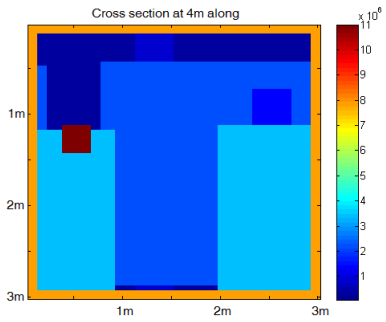

Figure 5.79: Cross section of the cargo container containing the small lead object a realistic example.

densities of each object, but what is interesting is that the case without lead has been increased more than that with it. Generally speaking the correct solutions for the case without lead have a slightly lower cost functionals than their counterparts in the case with lead, and the difference becomes more pronounced as the lead gets closer to the sensors, but in several examples the algorithm has adjusted the tolerance in the case without lead more. Since the same radial basis mesh has been used in both instances the only reason for this to happen would be the tolerance was increased too early without allowing enough time for the algorithm to decrease. This is one of the quirks of the genetic algorithm - at any point there could be a long period of time before another decrease in the cost functional occurs, then several will occur in quick succession. Deciding on a time scale which would indicate the algorithm having stalled is problematic and could be relatively arbitrary.

During the analysis of previous examples we began with a lens of size 40cm but here we begin with 50cm as this is the smallest lens which returns all positive results for both situations of lead being present and not (figures 5.82 and 5.83). It is only

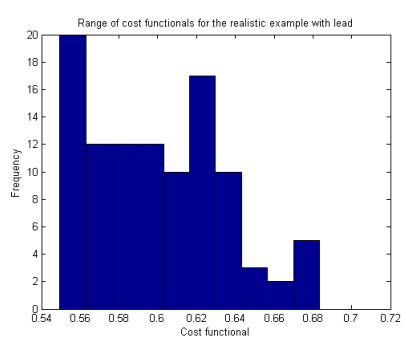

Figure 5.80: Histogram of the cost functionals when algorithm run for 5.79.

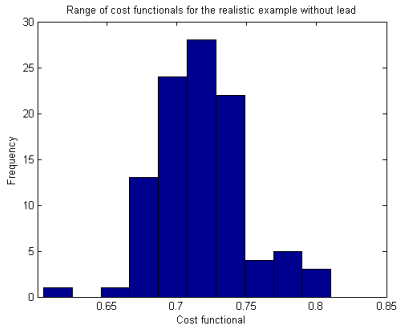

Figure 5.81: Histogram of the cost functionals when algorithm run for 5.79 without lead.

when the lens is increased to 60cm when some differences begin to show, but they are still not sufficient enough to provide us with a negative result in the case without lead (figures 5.84 and 5.85)

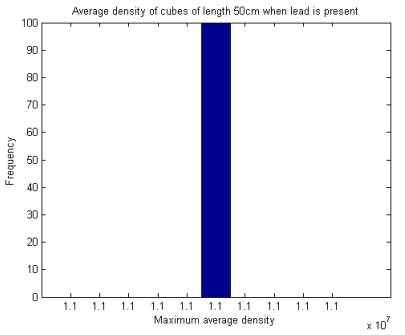

Figure 5.82: Histogram of the average density of cubes of size 50cm when using the data from 5.79.

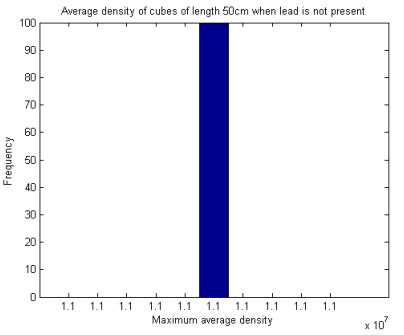

Figure 5.83: Histogram of the average density of cubes of size 50cm when using the data from 5.79 without lead.

Moving up to a lens of 70cm is when the results show a distinction. Figure 5.86 shows that in the case with lead almost all individuals contain an object close to the density of lead, whereas in figure 5.87 most individuals return a density less than 10g/cm^3 and so there has been a shift in this case. In the case where a lens of size 80cm was used the example with lead, figure 5.88, still returns a majority of individuals with a density greater than 10g/cm^3. Compare that to the instance without lead present, figure 5.89, and we see that all individuals have a density closer to the lower domain of D_3 rather than that of lead and so could be described as a negative result.

Both lens sizes of 70cm and 80cm seem to give us what we require for this example - correctly identifying lead when it is present and ignoring it when it is not. However a positive result could be whatever we define it to be. For the case where a lens of size

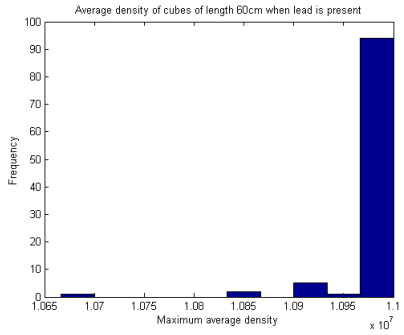

Figure 5.84: Histogram of the average density of cubes of size 60cm when using the data from 5.79.

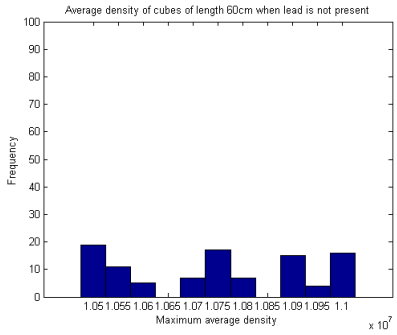

Figure 5.85: Histogram of the average density of cubes of size 60cm when using the data from 5.79 without lead.

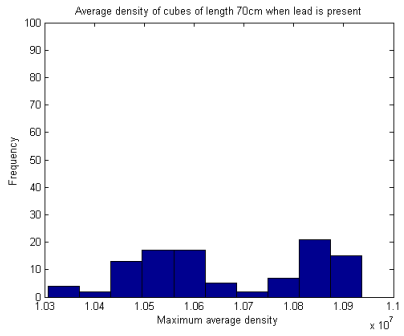

Figure 5.86: Histogram of the average density of cubes of size 70cm when using the data from 5.79.

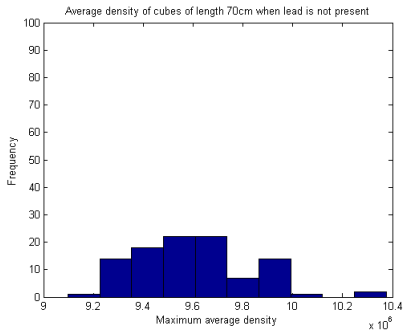

Figure 5.87: Histogram of the average density of cubes of size 70cm when using the data from 5.79 without lead.

90cm is used, figure 5.91 shows the results when lead is not present and, given that the individuals are centered around a density of 8g/cm^3, this is a negative result. The results from when lead is present in the domain is shown in figure 5.90 and this could be a borderline result.

More than half of the population have returned a density greater than 9.5g/cm^3. However it might be that we require more stringent rules for these cases. As we increase this threshold value the proportion of the population which achieves it will decrease and in this case it won't take much alteration to reach a point where less than half of the population reaches the threshold. Or we could require that more than half of the population is required to achieve the threshold value before it will be a positive result. These choices are explored in the next section.

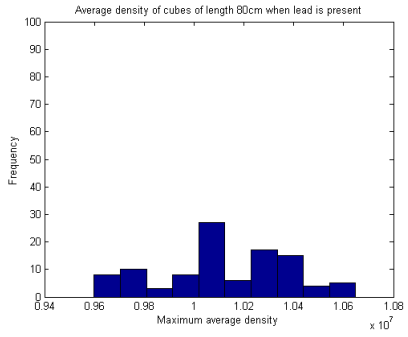

Figure 5.88: Histogram of the average density of cubes of size 80cm when using the data from 5.79.

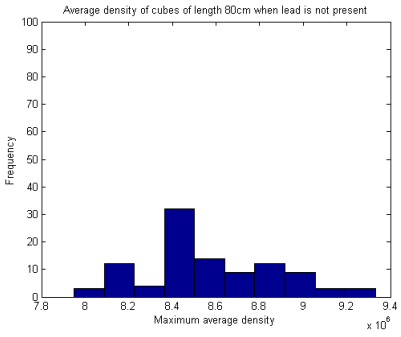

Figure 5.89: Histogram of the average density of cubes of size 80cm when using the data from 5.79 without lead.

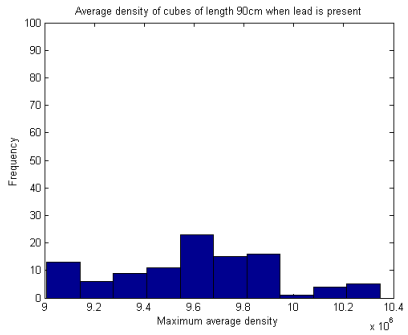

Figure 5.90: Histogram of the average density of cubes of size 90cm when using the data from 5.79.

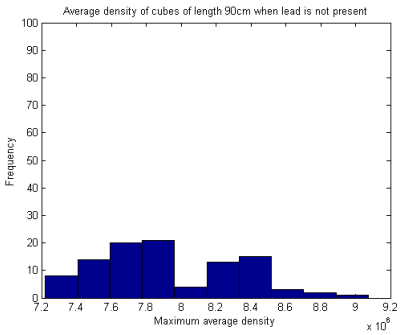

Figure 5.91: Histogram of the average density of cubes of size 90cm when using the data from 5.79 without lead.

5.3 Finding an indicator for the presence of fissile material

In the previous section 7 examples were chosen for their range of possibilities in order to contribute to our understanding of the problem, with the aim of finding a practical indicator of whether lead is present or not in a given cargo container. The next step is to attempt to collate all the information gleaned from the results so as to achieve this aim i.e. determine what size lens to use in any given scenario to provide us with correct positive and negative results. For this we need to restrict ourselves to what information we might know beforehand. It doesn't take much effort to measure the weight of the cargo container and so that can be used, as it has been mentioned and commented on throughout the previous section.

Another factor that has been mentioned is the size and location of the lead object. As it gets closer to a sensor location it has a noticeable effect on the data and therefore

the solutions obtained. Size may also contribute to this but since it is unlikely to have a very large lead object in the cargo container this was an aspect that was generally restricted to a maximum of around 40cm×40cm×40cm - here we have assumed that a small cube of fissile material of around 10cm on each side will require sufficient shielding to avoid setting off any radiation sensors. The location is less restricted as it is generally unknown. We might be able to say that we assume the fissile material to have been placed towards the center of the cargo container as an attempt to keep it away from as many sensors as possible, but it is not something that has been strictly assumed for this study. Instead certain allowances will have to be made for any such event. So the most obvious choice is to have the lens size be dependent on the weight of the cargo container.

Given that the tolerance chosen for the cost functional depended on the cargo container weight it might be possible that such a dependence has been imposed from the beginning. However that tolerance is treated as a starting value for the algorithm and is adjustable throughout when deemed necessary. The mechanism behind deciding when it is necessary could be improved upon but it is sufficient enough to provide viable results for analysis. Also an argument could be made that imposing such a relation is allowable if it leads to practically viable results.

However a simple relation such as this will not be sufficient to obtain the key lens sizes to be used for identification as we explored more than one instance in the previous section where a cargo container with a similar weight to another one did not share the same advisable lens size. In that case it was suggested that we look at the percentage of individuals over a prescribed density to act as a tolerance level, and so that is the first variable to consider for optimisation.

Our domain of interest, denoted by D_4, has a density of 11g/cm^3 and the domain closest to this is D_3 which has a density of 8g/cm^3. Midway between the two is a density of 9.5g/cm^3 which could be a good tolerance level to explore. However the results so far seem to indicate a preference towards false positives for several smaller lens sizes and so increasing the density tolerance slightly could be recommended. Also we have imposed these domains upon the reconstruction so choosing a tolerance level only based on that might not be the most accurate. This is the reasoning behind also exploring the use of 10g/cm^3 as a possible tolerance level.

The next step is to determine what percentage of individuals above this density would constitute a positive result. First just a straightforward majority would be the most obvious choice and so 50% is explored. However, due to the aforementioned preference towards false positives this may not be strict enough for our needs. Therefore larger values are also analysed. The results from this analysis are stored in tables 5.1 and 5.2. Explaining what they mean involves working backwards from the answer to figure out the question - another inverse problem.

For the moment we will focus on table 5.1. The seven examples have been listed with a letter for their description and the number they appear. Let us restrict ourselves to S1, which is the first simple example. We have already prescribed a density tolerance of 9.5g/cm^3, now we also prescribe a percentage tolerance of 50%. In this case if we use a lens of size up to 40cm for the data with lead present, more than 50% of the population return a maximum density of over 9.5g/cm^3, resulting in a true positive result. However if we instead look at the solutions created from the data without lead and use a lens of size 20cm, we also get more than 50% of the population indicating a maximum density over 9.5g/cm^3, which is a false negative. Therefore the only feasible lens sizes to use for this specific case are 20 or 30cm. For the rest of the examples n/a denotes that no answer was found.

	Feasible lens size (cm) for population %					
Example	50%	60%	70%	80%	90%	100%
S1	30-40cm	30-40cm	20-40cm	20cm	20cm	n/a
S2	50cm	n/a	n/a	30-40cm	30-40cm	20-30cm
SC1	60-70cm	60-70cm	60-70cm	60-70cm	60-70cm	60-70cm
SC2	60cm	60cm	50-60cm	50-60cm	50cm	50cm
C1	60cm	60cm	60cm	60cm	60cm	60cm
C2	n/a	n/a	n/a	n/a	n/a	n/a
R1	80-90cm	80-90cm	70-80cm	70-80cm	70-80cm	70-80cm

Table 5.1: Advised lens size to use to obtain correct positive and negative results when a population percentage tolerance is prescribed and the density tolerance is 9.5g/cm^3.

Looking at both tables the first obvious result is that both examples S2 and C2 have several instances where no answer was found. It is possible that both of these cases are special cases. For instance in the case of S2 the lead object was further towards the lower edge of the cargo container (figure 5.24), away from most sensors and so that could be the reason for the lack of results. Meanwhile for the case of C2

	Feasible lens size (cm) for population %					
Example	50%	60%	70%	80%	90%	100%
S1	30-40cm	30-40cm	20-40cm	20cm	20cm	n/a
S2	n/a	n/a	n/a	30-40cm	30-40cm	20cm
SC1	60-70cm	60-70cm	60-70cm	60-70cm	60-70cm	60cm
SC2	50-60cm	50-60cm	50cm	50cm	50cm	50cm
C1	60cm	60cm	60cm	60cm	60cm	50cm
C2	60cm	60cm	n/a	n/a	n/a	n/a
R1	70-80cm	70-80cm	70-80cm	70cm	70cm	70cm

Table 5.2: Advised lens size to use to obtain correct positive and negative results when a population percentage tolerance is prescribed and the density tolerance is 10g/cm^3.

the lead object is located towards the corner of the cargo container (figure 5.3), not necessarily very far away from the sensors. In one of the previous sections on sparsity (section 3.2.6) it was noted that in some instances it can be difficult to find objects in the corner of the cargo container, and it was suggested there was some unknown interaction between the sensors occurring. However this is not enough to completely disregard the two examples as special cases. Instead since in table 5.1 example C2 is unable to find even one feasible lens size let us focus solely on table 5.2. It might be difficult to spot a pattern and so the data has been rearranged into table 5.3 where it has been arranged in weight order and the columns are dependent on the lens sizes instead.

Using table 5.3 it is easier to spot a pattern linking the weight of the cargo container and a feasible lens size. Both of the simple examples could use a lens size anywhere between 20cm and 40cm, and they are very close in terms of weight so it should be allowable to group them together. Then there is a jump in weight to the semi-cluttered and cluttered examples which turn out to have very similar weights and so could also be grouped together. All four of these examples can use a lens of size 60cm to achieve good results. Moving on to the one realistic example a lens of size 70cm or 80cm would be feasible. Therefore it seems there is a general link between the weight of the cargo container and the size of lens we should use in the analysis of the results.

A problem may occur when attempting to decide on a percentage tolerance. Starting with the one realistic example we can choose a lens size of either 70cm or 80cm and then a percentage tolerance of 50-70% would cover both cases. However this is just one example, as we extend it to more realistic cases we might find different values

		Feasible percentage for lens size (cm)						
Weight (g)	Example	20cm	30cm	40cm	50cm	60cm	70cm	80cm
7.44×10^7	S1	70-90	50-70	50-70	n/a	n/a	n/a	n/a
7.45×10^7	S2	100	80-90	80-90	n/a	n/a	n/a	n/a
9.96×10^7	SC2	n/a	n/a	n/a	50-100	50-60	n/a	n/a
1.03×10^8	SC1	n/a	n/a	n/a	n/a	50-100	50-90	n/a
1.03×10^8	C2	n/a	n/a	n/a	n/a	50-60	n/a	n/a
1.1×10^8	C1	n/a	n/a	n/a	100	50-90	n/a	n/a
1.56×10^8	R1	n/a	n/a	n/a	n/a	n/a	50-100	50-70

Table 5.3: Advised population percentage to prescribe to obtain correct positive and negative results when a lens size is prescribed and the density tolerance is 10g/cm^3.

are better. Also this is the example which was excluded from the relation between the cargo container weight and the tolerance of the cost functional in order to see how it reacts to limited extrapolation. Adding this case to the data, along with other realistic cases, could help fine tune that relation to a point where it is more universal.

For the cases of SC1, SC2, C1 and C2 we could choose a lens size of 60cm along with a percentage tolerance of 50-60% and we would achieve good results. Three of these four cases are generally robust too, with either more than one lens size being feasible or a wider range of feasible percentage tolerance. However C2 has only one lens size that can be used with a relatively small range for the percentage tolerance. It could be that this is a very special case which wouldn't appear often, or it could be indicative of a wider range of solutions that will not fit into our proposed trend.

Alternatively the case of C2 could be an indication of a blind spot within the cargo container - a quirk based on the current set up. In comparison the realistic example contains a piece of lead fairly close to the edge, and possibly a sensor. There is a dependence not only on the sensor locations but also the structure within the cargo container. In this case it seems that lead in a more realistic example is easier to detect than in a more simple cluttered example. Many more examples would have to be explored to properly establish what eventuality is the most likely.

Finally are the two simple examples. We could use any lens size from 20cm to 40cm for these examples, but once a lens is chosen there is no feasible percentage tolerance shared by the two cases. For each of the three lens sizes we would require a larger percentage tolerance for case S2 than for S1. For now we will prescribe the lens size to be 30cm. Then the feasible percentage ranges for S1 and S2 are 50-70% and 80-90%

respectively. More importantly there are more individuals with false negatives in S2 than there are in S1 and so the answer must lie in the background density.

Below are two figures which show the location of the metal object in both cases with figure 5.92 relating to S1 and 5.93 relating to S2. The metal object is closer to the top of the cargo container in S2 than it is in S1, thereby contributing a larger gravitational gradient to the data and becoming more likely to be reconstructed as lead.

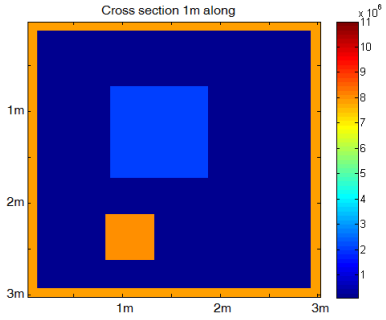

Figure 5.92: Location of the metal object for the first simple example.

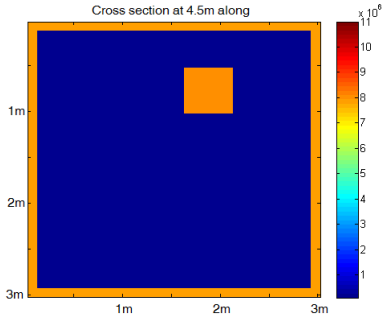

Figure 5.93: Location of the metal object for the second simple example.

There are other examples which have metal objects close to the sensors without having as much difference compared to those without and so the difference must be due to the sparse nature of the cargo container. Here the metal object (and also other objects) will have more of an affect on the data, creating a more wide range of possibilities without much difference in the weight. It might be possible to alter the algorithm to account for this in some way, but it could have adverse effects on the results for more cluttered examples. In a practical scenario these simple examples should be less likely to occur, and perhaps are more likely to be detected by other means. Therefore it would be most effective to keep the algorithm as it is.

5.4 Adding more sensors

The examples so far have all been performed under the assumption that there will be a restriction on the number of sensors able to be placed on the outer surface of the cargo container. It was stated in [51] that based on the size of commercially available gravitational sensors they could be safely spaced at around 1 meter apart, perhaps

slightly less. This motivated a choice of 4 sensors on each of the 3 edges of the sensor gate, totalling 12 sensors on the gate.

The cargo container would pass through this sensor gate, stopping at various points to take readings in order to build up sensory data covering three sides. Ideally we could stop the cargo container as many times as we want to take readings as the size of the sensors will have to influence on where the cargo container can be stopped. However another aim of this study is speed. The equipment is supposed to be constructed at a port where cargo containers are taken from ships and placed on lorries to be taken inland. The scanning needs to be performed as part of the movement from one to the other, ideally slowing down the transfer as much as possible.

Any time the cargo container moves we have to assume the contents will move too (we cannot rely on straps and other accessories being able to keep objects absolutely stationary). Moving objects would affect the readings and so to obtain the best results we would want to wait until all movement has ceased. Stopping the cargo container at more locations will result in longer waiting and therefore longer delays. Therefore the decision was made to simply apply the same restrictions in this direction too - readings taken 0.8m apart along the container. This resulted in a total of 96 sensor locations.

An argument could be made to increase the number of sensors. For instance sensors are improving, whether in accuracy or in a reduction in size, thereby negating some of the restrictions imposed. Alternatively instead of one sensor gate there could be several placed in a row, each one with sensors slightly offset from the gate before. By passing the cargo container through this series of gates it would be the equivalent of instead passing it through a single sensor gate with closely packed sensors. The result is many more sensor locations on the cargo container.

For instance let us say that the allowed distance between each sensor is 0.2m, whether this is realistically feasible or not. Each edge of the sensor gate will therefore have 15 sensors running along it. With 3 edges there will be a total of 45 sensors on the sensor gate. Allowing it to stop at 31 locations along the cargo container results in 1395 sensors placed on the outer edge of the cargo container, more than 10 times the number used before.

In order to compare this to the previous results it seems that choosing an example

from the previous section to work on would be the best course of action. So let us look at the first uncluttered example as shown in figure 4.1. This comprises of simply one object of lead, one of metal and one of wood surrounded by air. The algorithm was performed on both this setup and the equivalent setup without the lead object, thus comparing false positives versus false negatives.

The algorithm requires a cost functional tolerance to be prescribed beforehand - a type of target for the algorithm to reach towards. In the previous section this was approximated using the cargo container weight, with it being allowed to increase if the algorithm was failing to reach it over a large number of iterations. However changing the model results in that approximation becoming inaccurate. An increase in the number of data points means there are more individual elements of the discrete cost functional to sum up. For a solution deemed qualitatively close enough to the actual density distribution to be thought of as "correct" the cost functional will generally be higher.

For this example there were several density distributions created that were relatively close to the actual distribution - slight change in size and location of the three objects. The median of this range of cost functionals was used as an approximation to the cost functional tolerance. Then the algorithm was applied and ran until 100 solutions were found below this tolerance. The cost functionals of the solutions found for both cases with and without lead are shown below.

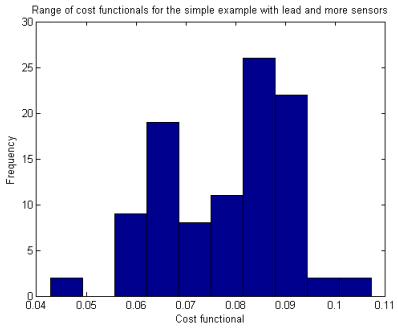

Figure 5.94: Histogram of the cost functional for more sensors on the first simple example.

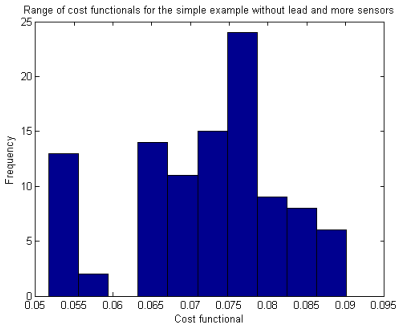

Figure 5.95: Histogram of the cost functional for more sensors on the first simple example without lead.

The ensemble algorithm is comprised of alternating between the steepest descent method and genetic algorithm, each one working on a number of individuals. The steepest descent algorithm is performed on 5 individuals for 20 iterations each with no

interaction between them. So it is given a starting density distribution (an individual) and applies the steepest descent algorithm for 20 iterations. Then it is given a second distinct density distribution (another individual) and applies the steepest descent algorithm for 20 iterations. By recording a density distribution at certain iterations for each of the 5 individuals we build a larger population of 30 individuals. This is then fed into the genetic algorithm where the 30 individuals interact at each iteration. 200 iterations of the genetic algorithm are applied.

The combination of the steepest descent algorithm and genetic algorithm totalling 220 iterations is called a "cycle". At each iteration the minimum cost functional is recorded and some of the cycles are plotted below in figures 5.96 to 5.98 for the case with lead. The case without lead produces similar figures.

Since the algorithm begins with randomly selected starting density distributions there is rapid reduction in the cost functional during the first cycle in figure 5.96. It is during the second cycle (figure 5.97) where the convergence becomes more steady. Figure 5.98 shows the convergence during the final five cycles, where the minimum cost functional jumps up each time an individual is sampled from the population to be added to the 100 solutions.

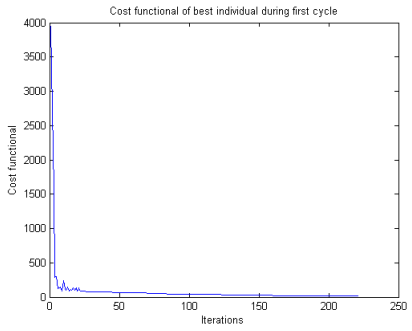

Figure 5.96: Cost functional for the first cycle of the first uncluttered case with more sensors.

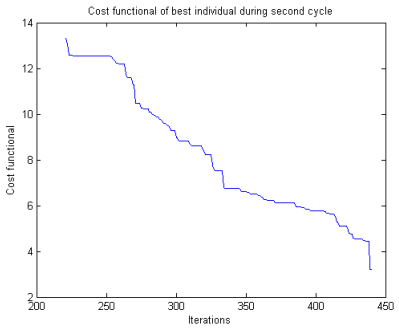

Figure 5.97: Cost functional for the second cycle of the first uncluttered case with more sensors.

As before the lens method is used to try to distinguish between the lead being present and not being present, histograms having been used to compare the two results in figures 5.99 to 5.106. The resulting histograms are presented in the same format as before, with those for lead present on the left, and those without lead on the right. The actual lead object is of size 25cm on each side. When an object of size 20cm is looked for, it is generally found in both cases - all solutions contain at least one lead

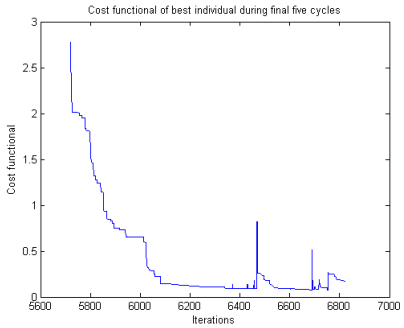

Figure 5.98: Cost functional for the final five cycles of the first uncluttered case with more sensors.

object of size 20cm. So respectively we get a true positive and a false positive.

Previously with fewer sensors on the same example we found a true positive for an object of size 20cm and around 70% of the population presenting a false positive in the absence of lead. Based on these preliminary results alone it would seem that, under this new sensor set up, lead is detected no matter what.

However, as move to those histograms involving larger lenses and comparing side by side, there is a difference between the two. In general there is a higher density reported for the case with lead than without, providing us with a possible distinction window with which to distinguish between false positives and false negatives. After analysing the results further it appears that a lens of size $30 - 40$cm on each side is a possible option with a threshold of 10g/cm^3. 95% of the population have an object of size 30cm with a density greater than 10g/cm^3 in the case with lead, as opposed to 58% in the case without lead. With a lens of size 40cm the population percentages are 81% and 45% respectively.

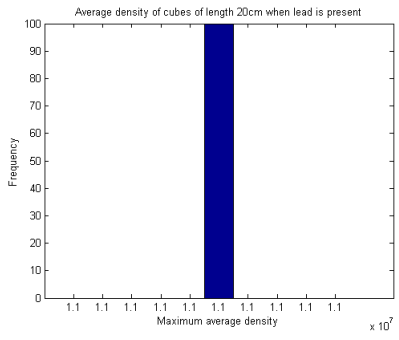

Figure 5.99: Histogram of the average density of cubes of size 20cm for more sensors on the first simple example.

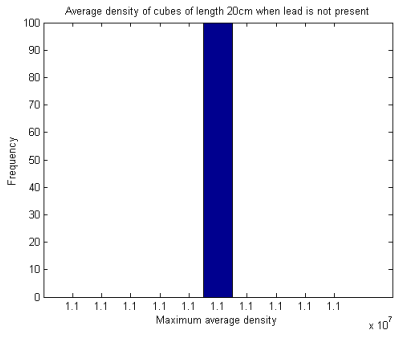

Figure 5.100: Histogram of the average density of cubes of size 20cm for more sensors on the first simple example without lead.

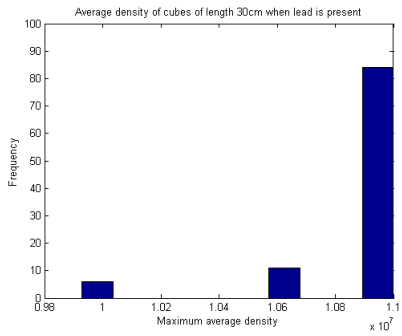

Figure 5.101: Histogram of the average density of cubes of size 30cm for more sensors on the first simple example.

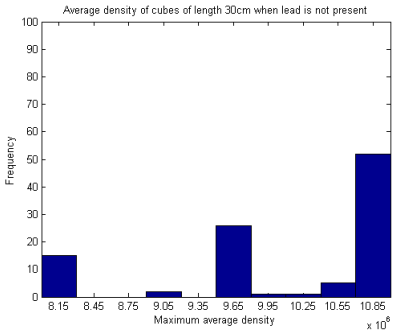

Figure 5.102: Histogram of the average density of cubes of size 30cm for more sensors on the first simple example without lead.

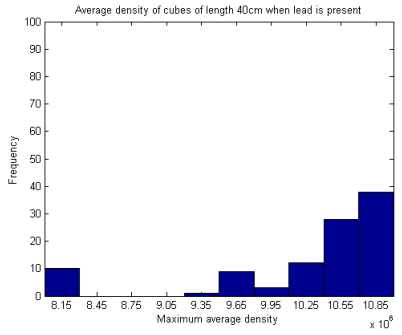

Figure 5.103: Histogram of the average density of cubes of size 40cm for more sensors on the first simple example.

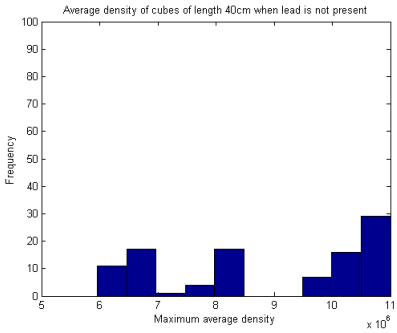

Figure 5.104: Histogram of the average density of cubes of size 40cm for more sensors on the first simple example without lead.

Given that the problem has been changed it is unsurprising that the distinction window has also changed. One might have expected that with more sensors there would be a greater distinction between the cases with and without lead. For the cluttered example the window has been slightly shifted, namely that a 20cm lens is no longer useable. This could imply a dependence on the sensor locations instead.

For comparison purposes, also included is another example from the previous section, that of the first cluttered example, for reasons that will soon become apparent. As before there are two histograms 5.107 and 5.108 showing the range of cost functionals in the end result. Also included are three plots of the evolution of the cost functional in figures 5.109 to 5.111 illustrating the behaviour at the start and towards the end of the algorithm in order to show convergence.

Figures 5.112 to 5.117 show the outcome when using the lens approach. It seems that for this case there is not a distinction window we can use to differentiate between

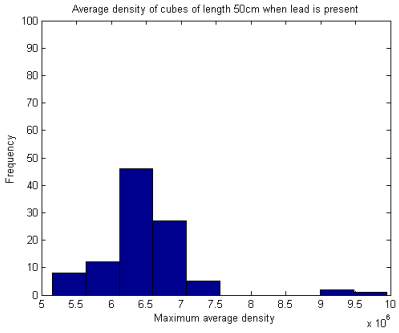

Figure 5.105: Histogram of the average density of cubes of size 50cm for more sensors on the first simple example.

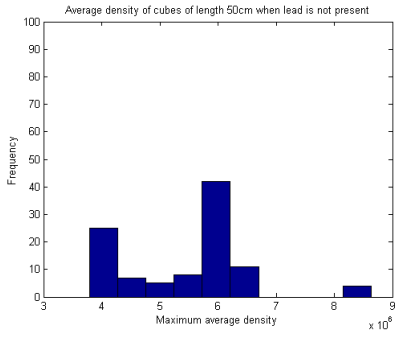

Figure 5.106: Histogram of the average density of cubes of size 50cm for more sensors on the first simple example without lead.

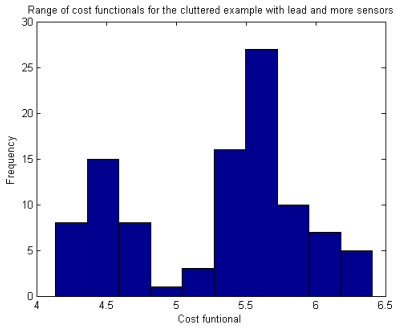

Figure 5.107: Histogram of the cost functional for more sensors on the first cluttered example.

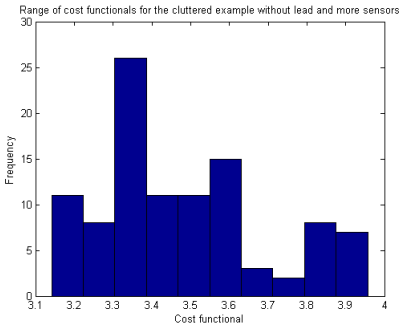

Figure 5.108: Histogram of the cost functional for more sensors on the first cluttered example without lead.

the two cases. Histograms for the presence of lead and without look generally similar, and in fact those without seem to return a higher density than those with lead. So the inclusion of more sensors has made the results less accurate in this case.

These results could mean several things. The first possibility is a simple failing of the algorithm. It was designed to attempt to reach a range of possible solutions, each of which is equally as valid in terms of cost functional but possibly very different. However what can happen is the algorithm simply chooses a similar solution over and over, resulting in 100 very similar solutions.

Certain restrictions were put into place to try and avoid this - the genetic algorithm is one. Another is the removal of many individuals from the population when a feasible solution is found. The assumption is that the feasible solution has exerted enough influence over the rest of the population to create several individuals very similar to itself. By removing more it should prompt the algorithm to find a suitably different

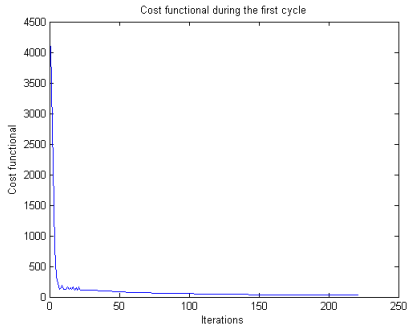

Figure 5.109: Cost functional for the first cycle of the first cluttered case with more sensors.

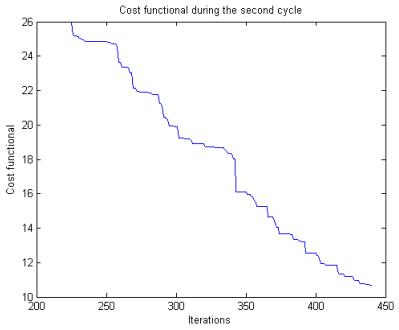

Figure 5.110: Cost functional for the second cycle of the first cluttered case with more sensors.

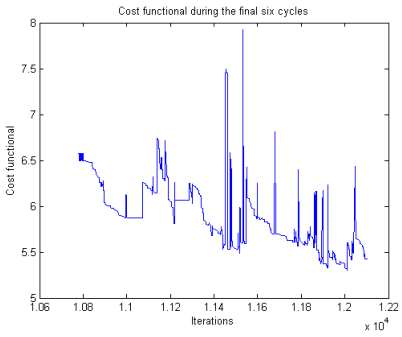

Figure 5.111: Cost functional for the final six cycles of the first cluttered case with more sensors.

feasible solution. So this should not be the reason. However much of the analysis was performed under the previous set up with fewer sensors. By adding more sensors we have changed the problem and therefore some decisions may no longer be viable.

Another possibility is a more heavy reliance on the sensor locations than previously thought. A lack of sensors could certainly create blind spots but perhaps using too many can lead to higher densities being favoured over lower ones. In the cluttered example shown above there is a metal object placed in between the lead object and the sensors, and it is located relatively close to the surface of the cargo container. After taking the lead away the metal box will still exert some influence over the sensors. As seen previously there are problem with evolution when an object is reconstructed close to the sensors. That was dealt with somewhat by moving the sensors further from the cargo container. However increasing the number of sensors could increase the problems involved.

It is possible there will be an optimum sensor arrangement, but another avenue of investigation could be the addition of select data. Each sensor does not necessarily need

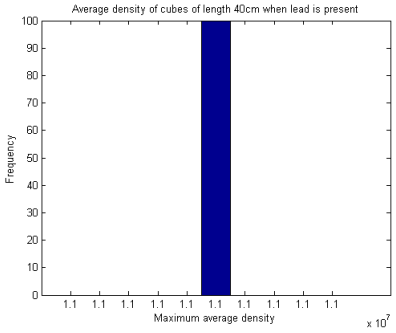

Figure 5.112: Histogram of the average density of cubes of size 40cm for more sensors on the first cluttered example.

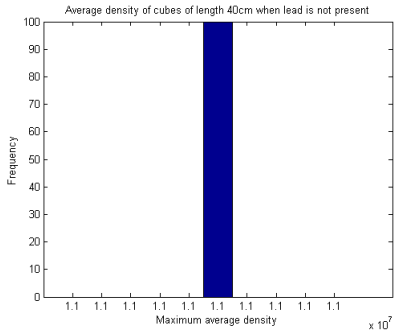

Figure 5.113: Histogram of the average density of cubes of size 40cm for more sensors on the first cluttered example without lead.

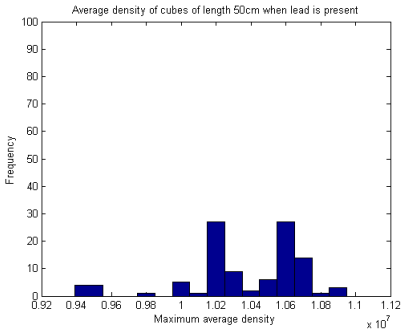

Figure 5.114: Histogram of the average density of cubes of size 50cm for more sensors on the first cluttered example.

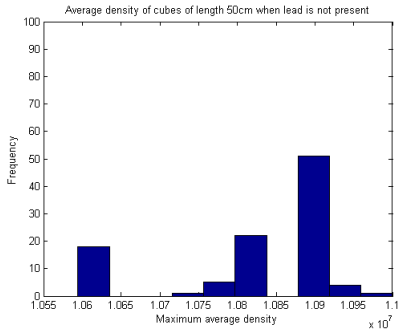

Figure 5.115: Histogram of the average density of cubes of size 50cm for more sensors on the first cluttered example without lead.

to contribute all 5 pieces of sensory data. It was found in [70] that some components of the gravity gradient tensor were better at detecting objects closer and others were better for further away. That analysis was performed for optimal design measures for larger length scales and so it may not be directly applicable here. A separate analysis would need to be performed.

It could be there is not a single optimal sensor arrangement, in which case several might have to be used in parallel. That is take several sets of data using different sensor arrangements and run the algorithm on each one separately to fins several sets of possible solutions which can then be compared/combined. However to come to such a conclusion would require much more analysis to be performed.

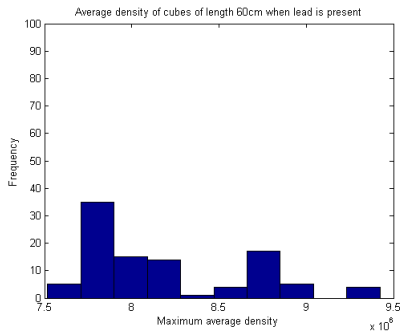

Figure 5.116: Histogram of the average density of cubes of size 60cm for more sensors on the first cluttered example.

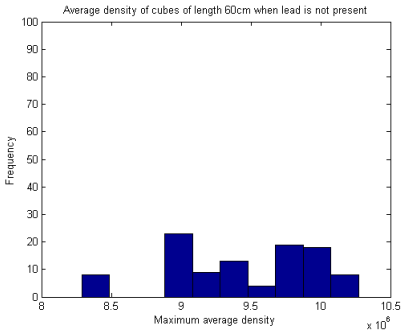

Figure 5.117: Histogram of the average density of cubes of size 60cm for more sensors on the first cluttered example without lead.

5.5 An ideal example

As shown so far the results can be unpredictable. Different set ups can completely change the results, and even the location of objects can heavily influence the outcome. As explored in the introduction the uniqueness of solutions has been studied. Once certain geometrical assumptions break down, so does a guarantee of uniqueness. The question is how much does the application of this algorithm help when those assumptions break down.

Uniqueness depends on having a single unknown object of unknown density surrounded by zero density. Instead we impose a domain where there are three objects of known density, one of lead, one metal and one wood. These are surrounded by zero density, so in this case the cargo container would be a vacuum with only three objects in it. Furthermore the three objects do not interact with each other - they exist as distinct entities within the cargo container in order to avoid certain possible influencing like that in the first cluttered example with more sensors (i.e. combinations of objects of different densities). The lead object is a cube of volume $25 \times 25 \times 25\text{cm}^3$.

The arrangement of sensors was that used in most of the results seen in this chapter - a total of 96 sensors on the surface of the cargo container. However the difference is that the same mesh grid was used for the data as for the reconstruction. Each voxel is a cube of size $10 \times 10 \times 10\text{cm}^3$ for both data and reconstruction. This prompts a slight change for the application of the algorithm. Previously the way in which an approximate feasible cost functional was found was by comparing a roughly correct reconstruction to the data directly. The subspace error from the use of level set

and radial basis functions was ignored and the cost functional treated as more of an approximate aim rather than set in stone. Following the same method would give us an aim of 0 for the cost functional, which is not reachable. So instead of allowing the algorithm to sample solutions as they get below a tolerance it is simply run until it is deemed to have stopped converging. This was performed on both the case with lead and without (the case without would therefore be only two objects surrounded by a vacuum - one of metal and one wood). The evolution of the cost functionals for both cases are shown in figures 5.118 and 5.119. In both cases the evolution has slowed down drastically. It may not have converged as it is difficult to decide on proper convergence for this application.

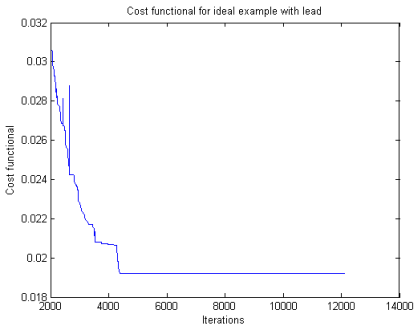

Figure 5.118: Cost functional in later iterations for the case with lead.

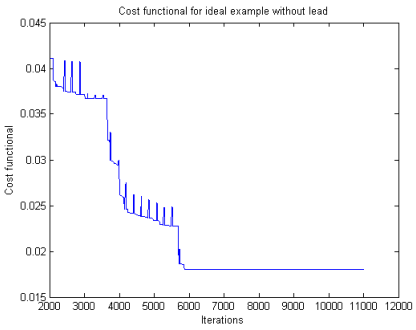

Figure 5.119: Cost functional in later iterations for the case without lead.

There is a further difference in the solutions associated with this approach. Previously a feasible solution was plucked from the population when it reached a certain tolerance and then replaced by others, eventually ending up with 100 possible solutions to analyse. To do so for the ideal case would take longer and would require changes to the code. However this case is designed to be more of a proof-of-concept and so 100 solutions is perhaps not required. Instead we made use of the state of the population at the end of the algorithm.

The population is of size 30 at the end - a result of the genetic algorithm. Many of the individuals may be relatively close to the correct solution but there will also be several kept an intentional distance apart - diversity is an integral part of the algorithm. Therefore all 30 certainly cannot be used. Instead we look at the 15 individuals with the lowest cost functionals. The histograms of both are shown in figures 5.120 and 5.121. There is a range of cost functionals for both but they have a similar upper

limit.

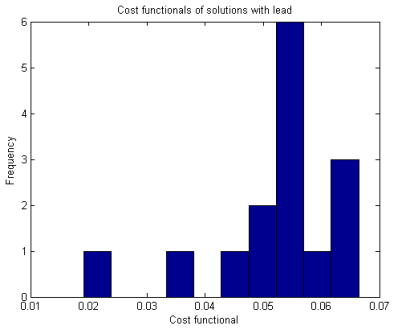

Figure 5.120: Cost functionals for 15 solutions for the case with lead.

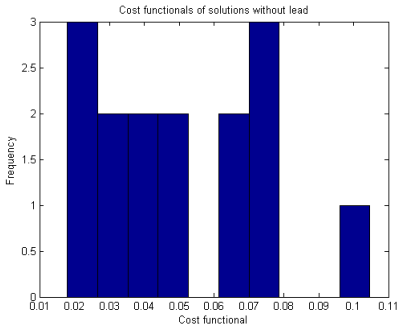

Figure 5.121: Cost functional for 15 solutions for the case without lead.

Following the same approach as before a lens was used of varying sizes to process the results and the histograms can be seen in figures 5.122 to 5.127. Those on the left are for the case with lead and those on the right are without. Focusing on the case with lead one can see that all solutions contain an object of size 20 which has the density of lead. Moving up to an object of size 30cm the histogram seems to suggest there are three solutions which contain an object of density 10g/cm^3 or above. Moving to 40cm the density has reduced to that of metal. This would imply that a few solutions contain a lead object slightly smaller than $30 \times 30 \times 30$cm^3 in the reconstruction, which is very close to the actual density distribution. It most likely exists on its own surrounded by some layer of vacuum - using a lens of size 40cm would encompass it and several voxels of zero density, resulting in a much lower average density than lead. However for this to be a good result it must be compared to the case without lead.

When a lens of size 20cm is used most of the solutions have a density very close to metal. The slight increase in density is due to the fact that at least one voxel remains for each domain, so there will always be a small amount of lead in the reconstruction. Here it appears that part of the metal object has a few voxels of lead within it. As we increase the lens size this gets smoothed out, resulting in most agreeing on a maximum density equal to that of metal.

It seems that for this case the algorithm has performed remarkably well. However it was chosen with this aim in mind, based on what we know about uniqueness and what the previous results have indicated. It may be that the algorithm works this well for all solutions of this rough structure, or there may be a few for which it fails. There

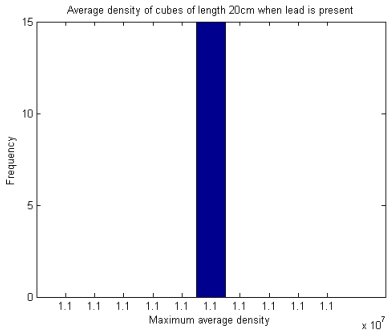

Figure 5.122: Histogram of the average density of cubes of size 20cm for the ideal example.

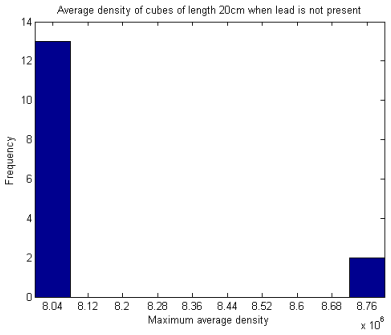

Figure 5.123: Histogram of the average density of cubes of size 20cm for the ideal example without lead.

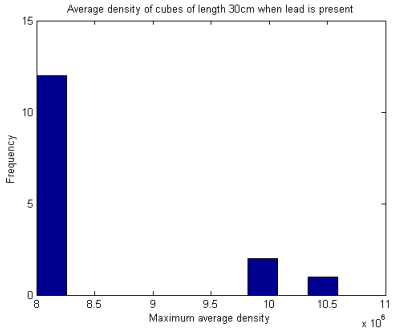

Figure 5.124: Histogram of the average density of cubes of size 30cm for the ideal example.

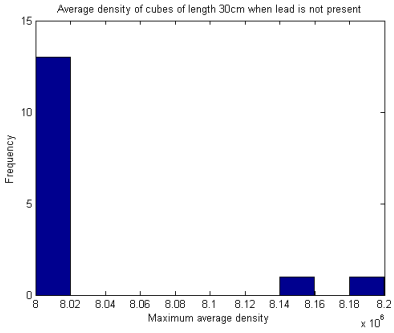

Figure 5.125: Histogram of the average density of cubes of size 30cm for the ideal example without lead.

is no guarantee either way.

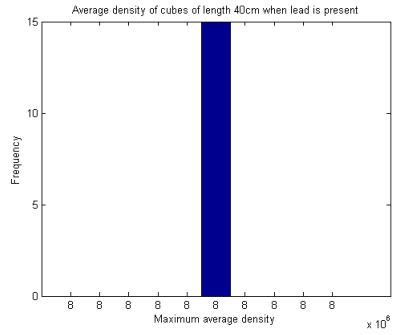

Figure 5.126: Histogram of the average density of cubes of size 40cm for the ideal example.

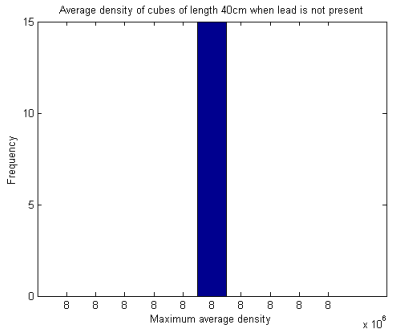

Figure 5.127: Histogram of the average density of cubes of size 40cm for the ideal example without lead.

Chapter 6

Summary and further work

6.1 Comparison to alternative methods

Use of gravity gradient data is prevalent in geophysical applications, usually for measuring deposits underneath the Earths surface. Several approaches have been considered throughout the years, and not all using level set techniques. For instance in the introduction the papers [59] uses level sets to find the density of a binary formulation, whereas [53] uses a genetic algorithm without the use of level set functions to the same degree of success. So level set functions are not necessarily the best approach for the general problem.

Indeed a more recent paper [71] advocates the use of a Tikhonov regularisation approach, with weighting functions to counteract the decay of sensitivities relative to the inverse distance cubed. The results indicate a minimum of 2 and sometimes 3 gravity gradient components to be available to obtain a good reconstruction of the density distribution. However this, much like most geophysical applications, are using a larger length scale. In particular [71] indicates using cells of size $50 \times 50 \times 20$m and taking measurements every 10m in a borehole.

To my knowledge the only paper that has attempted using gravity gradiometry along a shorter length is [51]. In that paper a Tikhonov regularisation approach is used for the specific purposes of finding fissile material. Weighting functions are used to counteract the problems involved with the inversion of gravity gradient data and there are some promising early results.

However the images created require some post-processing. Much of what was fissile material is recovered at a much lower density - the Tikhonov approach having smoothed out the reconstruction. However there is still a difference in the reconstructed density of the fissile material compared to the surroundings. A form of analysis known as funnel analysis is performed on the reconstruction to fins a threshold density above which is an indication of fissile material.

However while it finds a feasible threshold, it requires there to be a sufficient amount of fissile material for it to be feasible. Additionally it mentions the need to optimise the regularisation parameters involved to improve reconstruction, but also makes a point of stating that the reconstructions still suffer greatly from non-uniqueness of the solution. Therefore it leaves much open in terms of analysis.

I attempted to perform some of the analysis using Tikhonov regularisation during my MSc dissertation, which formed the groundwork for this project. As a brief summary I followed broadly the method used in the paper [51] - that is a Tikhonov regularisation approach. The parameters used were $\alpha_i = 1$ for $i = x, y, z$, the regularisation parameters relating to the gradients in the three coordinate directions. The parameter for magnitude was adjusted and explored in order to promote sparsity and was usually of the order of approximately 10^{-5}.

Additionally a Huber norm [42] was used to further promote sparsity, which is a combination of the 1-norm and 2-norm. The 1-norm is used for higher gradients, and the 2-norm used for lower gradients, with a switching point defined by a parameter θ, which also required optimisation. The results were varied and similar to those found in [51]. Generally speaking for a specific example the regularisation parameters could be optimised in order to obtain a good reconstruction (i.e. a distinct region of higher density compared to the surrounding area). However the optimised parameters were not universal - applying the same parameters to an alternative example would yield a completely smoothed out reconstruction, or one in which high density material appeared near the sensors. Weighting was used to counteract the sensitivity as in [71]. It was calculated using the values found in the sensitivity matrix $\mathbf{G}$, but it was not always successful.

Overall it was deemed that more analysis would be required to make a Tikhonov regularisation approach work, and there was no guarantee it would. Whereas the

theory seemed to suggest that by constraining the unknowns by using known densities we may be able to obtain better reconstructions. Therefore the level set method was used instead, but a Tikhonov approach could still be explored in the future. Or perhaps it could be incorporated into an ensemble algorithm to improve results.

6.2 Summary of the method

During this study we began by analysing the problem and finding it to be highly ill-posed with a large null space. A previous study was performed and published in [51] where Tikhonov regularisation was used in order to reach solutions closer to the actual density distribution. We decided to go in a different direction by exploring the use of level set functions with known prescribed densities in order to reduce the potential solutions we might reach in the null space.

The first avenue of exploration was the use of a gradient descent method. However it seemed that the restrictions we imposed meant that we were not achieving correct solutions. Instead the solutions would reconstruct a lead object whether it was actually present in the data or not. Relaxing some of the restrictions with regards to the initial guess would lead to false negatives instead of false positives and so it was determined that this method alone would not suffice.

Instead we focused on a genetic algorithm as such methods are able to explore the null space more freely, and therefore less likely to get stuck at a specific point as the gradient descent method would. The main downside was that the algorithm would take a long time to converge. Several alterations were made to increase the speed, such as the use of radial basis functions within the level set functions to reduce the number of unknown parameters. Unfortunately it was deemed too slow with too many false positives arising in the reconstructions.

Next we explored the use of sparse reconstructions. Methods that aim for sparse solutions would seem like a good fit as we are essentially looking for a spike in the density that would tell us whether to open the cargo container or not. Therefore the container was split into two distinct density distributions, one being sparse and the other being smooth. A method called Orthogonal Matching Pursuit was used on the sparse distribution and other methods (gradient-based and genetic) were used for the

smooth solution. However it was found that in order to find an accurate sparse solution it required an accurate smooth solution to work on beforehand, which is something neither method was likely to provide on its own.

In order to try and improve the results we returned to the gradient descent method, only with different restrictions added. Radial basis functions with fixed centers were used to keep away from the edges of the cargo container and the condition of a strict reduction in the cost functional was relaxed, with the aim of possibly escaping local minima. However a consequence of using the radial basis functions was creating new paths in which to get stuck, and the relaxation of the reduction was not enough to adjust it. While more alterations to the algorithm could have been made to avoid such local minima the choice was made to utilise this method in a different way.

Using the idea of combining methods together as a motivation we turned our attention to both the genetic algorithm and steepest descent method using level set functions incorporating radial basis functions. With the genetic algorithm being responsible for most of the convergence and the steepest descent exploring the local neighbourhood of several possible solutions we managed to create a population of solutions to analyse.

The analysis involves the optimisation of four parameters: a target cost functional $\mathcal{J}_{tol}$, a lens size with to process the solutions into quantifiable data, a target density which if an individual reached it would indicate a high density object present in said individual, and a target population percentage which would tell us to within some degree of certainty there is high density material in the actual density distribution.

Using the limited number of examples explored a general correlation between the cargo container weight and the lens size has been identified which could lead to an indicator for the presence of high density material within the cargo container. Verification of this correlation would require running the algorithm for a multitude of different types of examples in order to cover the whole range of possibilities.

It would also be recommended that a finer mesh be used in the reconstruction along with perhaps an increased number of sensors (up to a realistic maximum) in order to increase the accuracy of each solution. Based on the results from increasing the number of sensors a look into the sensor placement would also be required, with possibly multiple variations employed. This could also refine the relationship established between the cargo container weight and the prescribed tolerance $\mathcal{J}_{tol}$. It would

be a long-term project and computationally expensive but, along with some statistical analysis, should provide an overall success rate for the method developed throughout this study.

Bibliography

[1] D. Adalsteinsson and J. Sethian. A fast local level set function for propagating surfaces. *Journal of Computational Physics*, 106:77–91, 1993.

[2] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22, 1999.

[3] A. Aghasi, M. Kilmer, and E. L. Miller. Parametric level set methods for inverse problems. *SIAM J. Imaging Sciences*, 4(2):618–650, 2011.

[4] G. S. Alberti and H. Ammari. Disjoint sparsity for signal separation and applications to hybrid inverse problems in medical imaging. *Applied and Computational Harmonic Analysis*, 42(2):319–349, 2017.

[5] M. Azghani, P. Kosmas, and F. Marvasti. Microwave medical maging based on sparsity and an iterative method with adaptive thresholding. *IEEE Transactions on Medical Imaging*, 34(2):357–365, 2015.

[6] P. Baesso, D. Cussans, C. Thomay, and J. Velthius. Toward a rpc-based muon tomography system for cargo containers. *Journal of Instrumentation*, 9, 2014.

[7] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[8] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[9] R. E. Bell. Gravity gradiometry. *Scientific American*, 278(6):74–79, 1998.

[10] J. M. Bioucas-Dias and M. T. Figueiredo. A new twist: two-step iterative-shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, 2007.

[11] T. B. Blackwell and V. A. Kudryavtsev. Identification of nuclear materials in cargo containers using cosmic rays. *IEEE*, 2013.

[12] K. Bredies, D. A. Lorenz, and P. Maass. A generalized conditional gradient method and its connection to an interative shrinkage method. *Computational Optimization and Applications*, 42(2):173–193, 2009.

[13] G. Brown. A bayesian method for imaging voids using gravimetry. *IMA Conference on Mathematics in Defence*, 2015.

[14] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[15] M. D. Buhmann. Radial basis functions. *Acta Numerica*, Acta Numerica 9., Cambridge University Press:1–38, 2000.

[16] M. D. Buhmann. Radial basis functions : theory and implementations. *Cambridge monographs on applied and computational mathematics*, 12, 2003.

[17] M. Burger. A level set method for inverse problems. *Inverse Problems*, 17(5):1327–1355, 2001.

[18] E. J. Candes and F. Guo. New multiscale transforms, minimum total variation synthesis: applications to edge-preserving image reconstruction. *Signal Processing*, 82(11):1519–1543, 2002.

[19] D. U. Carlos, M. A. Braga, H. F. Galbiatti, and W. R. Periera. Airborne gravity gradiometry - data processing and interpretation. *Revista Brasileira de Geofisica*, 31(3):428–452, 2014.

[20] T. Cecil, J. Qian, and S. Osher. Numerical methods for high dimensional hamilton-jacobi equations using radial basis functions. *Journal of Computational Physics*, 196(1):327–347, 2004.

[21] A. Chambolle, R. A. Devore, N. yong Lee, and B. J. Lucier. Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing*, 7(3):319–335, 1998.

[22] P. Charbonneau. An introduction to genetic algorithms for numerical optimization. *Technical Report*, 2002.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[24] D. L. Chopp. Another look at velocity extensions in the level set method.

[25] D. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory*, volume 93 of *Applied Mathematical Sciences*. Springer, 1992.

[26] I. Daubechies, M. Defrise, and C. D. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commuications on Pure and Applied Mathematics*, 57:1413–1457, 2004.

[27] D. DiFransesco, T. Meyer, A. Christensen, and D. FitzGerald. Gravity gradiometry - today and tomorrow. *11th SAGA Biennial Technical Meeting and Exhibition*, pages 80–83, 2009.

[28] D. L. Donoho. Nonlinear solution of linear inverse problems by wavelet-vaguelette decomposition. *Applied and Computational Harmonic Analysis*, 2:101–126, 1995.

[29] O. Dorn and D. Lesselier. Level set methods for inverse scattering. *Inverse Problems*, 22(4):R67–R131, 2006.

[30] C. P. Dubey and V. M. Tiwari. Computation of the gravity field and its gradient: some applications. *Computers and Geosciences*, 88:83–96, 2016.

[31] M. I. Estifeev. The state of the art in the development of onboard gravity gradiometers. *Gyroscopy and Navigation*, 8(1):68–79, 2016.

[32] C. L. M. et al. Tomographic imaging with cosmic ray muons. *Science and global security*, 16:37–53, 2008.

[33] L. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 1998.

[34] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[35] A. Gelas, O. Bernard, D. Friboulet, and R. Prost. Compactly supported radial basis functions based collocation method for level-set evolution in image segmentation. *IEEE Transactions on Image Processing*, 16(7):1873–1887, 2007.

[36] P. Getreuer. Rudin-osher-fatemi total variation denoising using split bregman. *Image Processing On Line*, 2:74–95, 2012.

[37] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.

[38] P. Gonzales-Rodriguez, M. Kindelan, M. Moscoso, and O. Dorn. History matching problem in reservoir engineering using the propagation-backpropagation method. *Inverse Problems*, 21(2):565–590, 2004.

[39] H.-M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, 2001.

[40] F. Hettlich and W. Rundell. Iterative methods for the reconstruction of an inverse potential problem. *Inverse Problems*, 12:251–266, 1996.

[41] P. Houghton, P. Nuttall, M. Cvetkovic, and S. Mazur. The role of potential fields as an early dataset to improve exploration in frontier areas. *first break*, 32:79–85, 2014.

[42] P. J. Huber. Robust statistical procedures. *SIAM*, 68(67), 1996.

[43] N. Irishina, D. Alvarez, O. Dorn, and M. Moscoso. Structural level set inversion for microwave breast screening. *Institute of Physics Publishing, Inverse Problems*, 26:1–26, 2010.

[44] V. Isakov. *Inverse Source Problems*, volume 34 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, Rhode Island, 1990.

[45] V. Isakov. *Inverse Problems for Partial Differential Equations*, volume 127 of *Applied Mathematical Sciences*. Springer, 2 edition, 2006.

[46] V. Isakov, S. Leung, and J. Qian. A fast local level set method for inverse gravimetry. *Communications in Computational Physics*, 10(4):1044–1070, 2011.

[47] B. Jin, T. Khan, and P. Maass. A reconstruction algorithm for electrical impedance tomography based on sparsity regularization. *International Journal for Numerical Methods in Engineering*, 89(3):337–353, 2012.

[48] A. Kadu, T. van Leeuwen, and W. A. Mulder. Salt reconstruction in full waveform inversion with a parametric level-set method. *IEEE Transactions on Computational Imaging*, 3(2):305–315, 2017.

[49] J. Kaipio and E. Sommersalo. Statistical inverse problems: discretization, model reduction and inverse crimes. *Journal of Computational and Applied Mathematics*, 198:493–504, 2007.

[50] B. Kirkendall, P. Harben, and P. Lewis. Advances of crosswell electromagnetics in steel casing. *Society of Exploration Geophysicists*, 54(2):323–326, 1999.

[51] B. Kirkendall, Y. Li, and D. Oldenburg. Imaging cargo containers using gravity gradiometry. *IEEE Transactions on Geoscience and Remote Sensing*, 45:1786–1797, 2007.

[52] I. Knowles. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148(1):2–22, 1999.

[53] R. A. Krahenbuhl and Y. Li. Inversion of gravity data using a binary formulation. *Geophysical Journal International*, 167:543–556, 2006.

[54] Y. Li and D. W. Oldenburg. 3d inversion of gravity data. *Geophysics*, 63(1):109–119, 1998.

[55] Y. Li and D. W. Oldenburg. Fast inversion of large-scale magnetic data using wavelet transforms and a logarithmic barrier method. *Geophysical Journal International*, 152(2):251–265, 2003.

[56] W. Lin and M. Zhdanov. 3d inversion of gravity and gravity gradiometry data using multinary transformation of the model parameters. *SEG International Exposition and 86th Annual Meeting*, pages 1516–1520, 2016.

[57] A. Litman, D. Lesselier, and F. Santosa. Reconstruction of a two-dimensional binary obstacle by controlled evolution of a level-set. *Inverse Problems*, 14(3):685–706, 1998.

[58] J.-C. Liu. An inverse source problem of the poisson equation with cauchy data. *Electronic Journal of Differential Equations*, 2017(119):1–19, 2017.

[59] W. Lu and J. Qian. A local level set method for three-dimensional inversion of gravity gradient data. *Geophysics*, 80:1, 2015.

[60] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[61] M. S. McMillan, C. Schwarzback, E. Haber, and D. W. Oldenburg. 3d parametric hybrid inversion of time-domain airborne electromagnetic data. *Geophysics*, 80(6):K25–K36, 2015.

[62] V. Michel and A. S. Fokas. A unified approach to various techniques for the non-uniqueness of the inverse gravimetric problem and wavelet-based methods. *Inverse Problems*, 24:1–24, 2008.

[63] V. Michel and S. Orzlowski. On the null space of a class of fredholm equations of the first kind. *Journal of Inverse Ill-Posed Problems*, 24(6):687–710, 2016.

[64] N. Naik, R. Beatson, J. Eriksson, and E. van Houten. An implicit radial basis function based reconstruction approach to electromagnetic shape tomography. *Inverse Problems*, 25(2), 2009.

[65] J. Neuberger. Sobolev gradients and differential equations. 1997.

[66] J. Nocedal and S. J. Wright. Numerical optimization. *New York: Springer-Verlag*, 1999.

[67] M. Okabe. Analytical expressions for gravity anomalies due to homogeneous polyhedral bodies and translations into magnetic anomolies. *Geophysics*, 44(4):730–741, 1979.

[68] S. Osher and R. P. Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational Physics*, 169(6):463–502, 2001.

[69] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *The 27th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA*, 1:40–44, 1993.

[70] M. Pilkington. Analysis of gravity gradiometer inverse problems using optimal design measures. *Geophysics*, 77(2):G25–G31, 2012.

[71] H. Rim and Y. Li. Single-hole imaging using borehole gravity-gradiometry. *Geophysics*, 77(5):G67–G76, 2012.

[72] A. H. Saad. Understanding gravity gradients. *SEG Houston conference paper*, pages 643–646, 2005.

[73] F. Santosa. A level set approach for inverse problems involving obstacles. *Control, Optimisation and Calculus of Variations*, 1:17–33, 1996.

[74] M. K. Sen and P. L. Stoffa. Global optimization methods in geophysical inversion. *Cambridge University Press*, 2013.

[75] J. A. Sethian. A marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93:1591–1595, 1996.

[76] J. A. Sethian. Fast marching methods. *SIAM Review*, 41:199–235, 1999.

[77] G. G. Stokes. On the internal distribution of matter which shall produce a given potential at the surface of a gravitating mass. *Proceedings of the Royal Society of London*, 15:482–486, 1866-1867.

[78] C. Thomay, J. Velthius, P. Baesso, D.Cussans, C. Steer, J. Burns, and S. Quillan. A novel technique to detect special nuclear material using cosmic rays. *Nuclear Science Symposium and Medical Imaging Conference Record*, 2012.

[79] A. V. Veryaskin. Gravity, magnetic and electromagnetic gradiometry. *IOP Concise Physics*, 2018.

[80] L. Wan and M. S. Zhdanov. Iterative migration of gravity and gravity gradiometry data. *SEG Houston 2013 Annual Meeting*, pages 1211–1215, 2013.

[81] S. Wang and M. Y. Wang. Radial basis functions and level set method for structural topology optimization. *International Journal for Numerical Methods in Engineering*, 65(12):2060–2090, 2006.

[82] S. xin Zhu. Compactly supported radial basis functions: how and why. *Oxford University Research Archive*, 2012.

[83] U. Yurtsever. On the gravitational inverse problem. *Applied Mathematical Sciences*, 5(57):2839–2854, 2011.

[84] M. S. Zhdanov. New advances in regularized inversion of gravity and electromagnetic data. *Geophysical Prospecting*, 57:463–478, 2009.

[85] M. S. Zhdanov. 3d potential field migration for rapid imaging of gravity gradiometry data - a case study from broken hill, australia, with comparison to 3d regularised inversion. *SEG San Antonio 2011 Annual Meeting*, pages 825–829, 2011.