Expansion of Regularization Tools with Large-Scale Problems

Agnes Martine Nielsen & Astrid Enslev Vestergård

Kongens Lyngby 2013
B.Sc.-2013-9

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Matematiktorvet, Bygning 303 B, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253031, Fax +45 45881399
compute@compute.dtu.dk
www.compute.dtu.dk B.Sc.-2013-9

Summary

The goal with this expansion pack for Regularization Tools has been to create good, interesting, and easy to use large-scale test problems for testing numerical algorithms in Matlab. We have implemented three different kind of test problems: *1*) Gravity Surveying in 2-D and 3-D; *2*) Seismic Tomography in 2-D and 3-D with Fresnel Kernels; and *3*) Image deblurring problem. In the implementation and manual we have focused on consistency and structure. This means, that all the functions have the same sort of inputs and outputs with few variations. This should make it easier for the user to use the different test problems. User-friendliness has also been a focus point for us. We have achieved this by creating functions that require very few input parameters, but yet still can be adjusted to a specific need by the advanced user. All the geology problems have predefined examples, which makes it easy to get started with them. The image deblurring problem has several standard images that serve in the same way.

Another common thing for all the functions, is that there is a detailed Matlab interface available, as well as this manual and demo scripts. We imagine that many of the users will be using personal computers to test their numerical algorithms. We have therefore sought to make the code as efficient as possible as well as using sparse matrices, when it was suitable, to minimize the required memory space.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring an B.Sc. in Engineering. The work has been carried out from February to June 2013 under supervision of Professor Per Christian Hansen.

We would like to thank Per Christian Hansen for always being available for answering our questions and for the comprehensive weekly meetings.

We hope you will find this manual useful, when using our Matlab package.

Lyngby, 14-June-2013

Agnes Martine Nielsen & Astrid Enslev Vestergård

List of Symbols

$\mathbf{A}$	The system matrix
$\mathbf{b}$	The right-hand side
$\mathbf{x}$	The exact solution to the inverse problem
$\overset{\frown}{\square}$	The 180 degrees rotation of $\square$
$\det(\cdot)$	The determinant
$\|\cdot\|_2$	The Euclidean norm
$\mathrm{vec}(\cdot)$	Vectorization of matrix
$\min(\cdot,\cdot)$	Minimum of two elements

Contents

Chapter 1

Introduction

Numerical analysis is an important part of modern computer science. It is used to approximate or recreate solutions of mathematical systems that cannot be solved analytically. There are many software systems and packages available that implement different numerical algorithms. One of these, is the Matlab package Regularization Tools [4], for which this is an expansion pack. Regularization Tools is a package for analyzing and solving discrete ill-posed problems. This expansion pack adds several test problems to the package, these are designed for testing numerical algorithms that solve discrete linear inverse problems. An inverse problem is: When we know the output of a system, but we only know either the system or the exact input and want to find the unknown of the two, see Figure 1.1. If both the input and the system are known, it is on the other hand called a forward problem. The test problems are formulated as forward problems, so the system and input are set up and then the output is computed. The problems are given to the user to solve as inverse problems, but where the exact solution is also known. The user can then compare the solution they find with the given exact solution. This is important when designing an algorithm, for example regularization algorithms, since it is a way to evaluate the performance of the algorithm. Comparing the solution also shows the need for test problems, since we can never get the exact solution to use for evaluation when working with real data. With test problems, it is possible to generate all three parts of an inverse problem, that is, the exact data, which is the input, the system, and the output and compare the computed solution to the exact

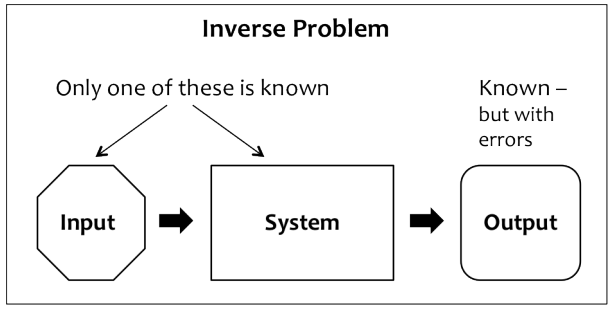


Figure 1.1: The Inverse Problem.

solution.

Test problems can in general be used for several purposes, two of these are checking whether a physical modeling is done correctly, the other is to test algorithms. We have focused on the latter in this expansion pack, therefore the physical models used are simplified and the parameters may not reflect real physical parameters. The test problems are however designed to reflect the size of real life problems. This means that all physical modeling is done in two or three dimensions, which results in very large systems. This is opposed to the test problems already implemented in Regularization Tools, which are from a time, where systems of this size were too hard to solve. When dealing with these large-scale systems there is still a problem of storing the entire system, therefore minimizing the needed memory space has been a focus in the designs.

The discrete linear problem can be written as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where $\mathbf{A}$ is the system matrix that transforms the exact data, $\mathbf{x}$, to the measured data, $\mathbf{b}$. The test problems in this expansion pack will all be of this form, that is, they are all discrete linear inverse problems. These linear inverse problems can however arise from a variety of different real life problems. Some examples are: Recreating a sharp image from a blurred image or recreating the mass distribution in the underground in an area given gravity measurements on the surface. These are also two of the three cases implemented as test problems, the third is a problem of recreating an image of the subsurface using seismic waves.

Chapter 2

Gravity Surveying Problems

In gravity surveying we measure differences in the vertical gravity field on the surface of the earth and use the measurements to reconstruct the geometry of the mass densities in the subsurface. This helps geologists understand what is in the underground. This problem is an inverse problem of the kind in equation (1.1), and further reading about the topic can be found in [2].

In this section, we will consider a two dimensional version of the problem based on sections 2.1 and 2.2 in [5]. We need a model that, given the mass density distribution in the underground, can compute the gravitational field. From physics we know that the gravitational field due to a point mass is the vector

$$\mathbf{g} = -GM\frac{\hat{\mathbf{r}}}{\|\mathbf{r}\|_2^2}, \tag{2.1}$$

where G is the gravitational constant, M is the strength of the point mass, $\mathbf{r}$ is the vector from the measuring point to the point mass, and $\hat{\mathbf{r}}$ is the unit vector: $\hat{\mathbf{r}} = \mathbf{r}/\|\mathbf{r}\|_2$. For our purpose, we will modify the model slightly. Firstly, we will neglect the gravitational constant and reverse the vertical axis to get rid of the minus sign. Reversing the axis means that we get a left-hand coordinate system. Secondly, we are not only interested in one point mass, but many point masses so we need to integrate over the subsurface.
We imagine that we take a slice of earth so that we have a two dimensional representation of the underground as shown in Figure 2.1.

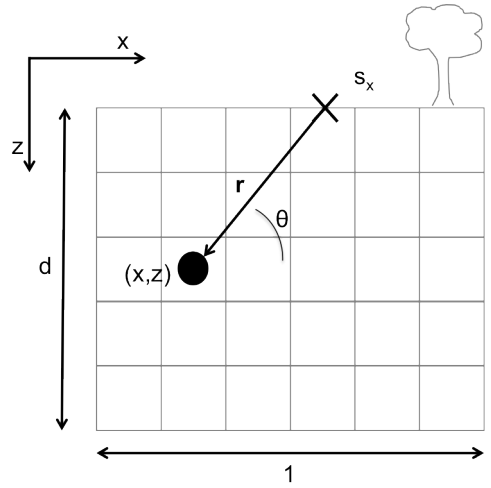


Figure 2.1: Geometry of two dimensional gravity surveying problem.

Given the geometry in the underground, we have to compute the vertical gravity field at a point, s, on the surface. We consider the contribution to the gravity field, dg, from a source point, (x, z), below the surface. We let the mass density be a function of the position, so at the point (x, z) it is $f(x, z)$. The distance between the measuring point s and the source point (x, z) is $\|\mathbf{r}\|_2 = \sqrt{z^2 + (s-x)^2}$. Since we are only interested in the vertical component of the gravitational force, we will only use the vertical component of the unit vector $\hat{\mathbf{r}} = [-\cos(\theta), \sin(\theta)]^T$. That is, $\sin(\theta) = \frac{z}{\|\mathbf{r}\|_2}$, where θ is the angle in radians as shown in Figure 2.1. We can insert this in equation (2.1),

$$dg = \frac{\sin(\theta)}{\|\mathbf{r}\|_2^2} f(x,z) = \frac{z}{\|\mathbf{r}\|_2^3} f(x,z) = \frac{z}{(z^2 + (s-x)^2)^{3/2}} f(x,z).$$

We notice that $f(x, z)$ in the equation above is the mass density (and not the mass). The mass distribution is found by integrating over the subsurface. The vertical component of the gravitational field at a point s then becomes,

$$g(s) = \int_0^1 \int_0^d \frac{z}{(z^2 + (s-x)^2)^{3/2}} f(x,z) dz dx. \tag{2.2}$$

The integration is from 0 to 1 along the x-axis and from 0 to the depth, d, along the z-axis. Equation (2.2) is a first kind Fredholm integral, where the kernel is

$$K(s,x,z) = \frac{z}{(z^2 + (s-x)^2)^{3/2}}.$$

We are however not interested in a continuous representation of the model, but a discrete one.
We discretize the area into quadratic pixels with N_x pixels in the x-direction and N_z pixels in the z-direction. We use the midpoint discretization method from [8] and represent x and z as follows,

$$x_l = \frac{l - \frac{1}{2}}{N_x}, \quad z_k = \frac{(k - \frac{1}{2})d}{N_z}, \quad l = 1 \dots N_x, k = 1 \dots N_z,$$

where l is the index in the x-direction and k is the index in the z-direction of the subsurface. This gives a kernel that depends on the indices,

$$K(s, x, z) \approx K_{l,k}(s) = \frac{z_k}{(z_k^2 + (s - x_l)^2)^{3/2}}.$$

This leads to the following discretization of the integral (2.2),

$$\int_0^1 \int_0^d K(s) f(x, z) dz dx \approx \int_0^1 \frac{d}{N_z} \sum_{k=1}^{N_z} K_k(s) f(x, z_k) dx, \tag{2.3a}$$

$$\approx \frac{1}{N_x} \frac{d}{N_z} \sum_{l=1}^{N_x} \sum_{k=1}^{N_z} K_{l,k}(s) f(x_l, z_k), \tag{2.3b}$$

$$= \frac{d}{n} \sum_{l=1}^{N_x} \sum_{k=1}^{N_z} K_{l,k}(s) f(x_l, z_k) = \tilde{g}(s), \tag{2.3c}$$

where $n = N_x \cdot N_z$. We discretize the measurement interval into m measurement points, s_x, and let $b_i = \tilde{g}(s_x)$ for $i = 1 \dots m$. We can now formulate the problem as the system (1.1), that is,

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{x}$ is a vector of the $f(x, z)$ values in each pixel and this is called the phantom. The elements of $\mathbf{A}$ are given as,

$$a_{i,j} = \frac{d}{n} K_{l,k}(s_x),$$

where $j = k + (l - 1)N_z$.

2.1 geophantoms

Purpose:

Create exact material distributions for geological test problems.

Synopsis:

```
X = geophantoms(N)
X = geophantoms(N,example)
X = geophantoms(N,example,options)
```

Description:

Input Parameters	
N	A vector of length 2 or 3, depending on the desired dimensions that defines the size of the phantom **X**. The form should be $\mathbf{N} = [N_x\ N_z]$ if 2-D or $\mathbf{N} = [N_x\ N_y\ N_z]$ if 3-D.
example	Example number indicating the chosen example. Possible examples are 1, 2, 3, 4, or empty. If empty, the phantom specified by the option *parameters* is created.
options	Struct containing the option fields for the test problem, see below.

Output Parameters	
X	Phantom matrix of dimensions specified in **N**.

Option Fields	
parameters	A vector with the parameters of the wanted phantom. The first element specifies the type. It can be 1 for a disk/ball/ellipse/ellipsoid, 2 for a Gaussian object or 3 for a box. In 2-D the parameters for the disk/ellipse are: $[1, x_0, z_0, a_1, a_2, theta]$, where (x_0, z_0) is the center, a_1 and a_2 are the semi-axes and *theta* is the rotational angle in radians. If $a_1 = a_2$ the object is a disk. The parameters for the 2-D Gaussian function are: $[2, mu_1, mu_2, sigma_1, sigma_2, cor]$, where (mu_1, mu_2) is the center, $sigma_1$ and $sigma_2$ are the standard deviations and *cor* is the correlation.

	The parameters for the 2-D box are: $[3, x_0, z_0, a_1, a_2]$, where (x_0, z_0) is the center and a_1 and a_2 are half the lengths of the sides of the box. In 3-D the parameters for the ball/ellipsoid are: $[1, x_0, y_0, z_0, a_1, a_2, a_3, theta_1, theta_2, theta_3]$, where (x_0, y_0, z_0) is the center and a_1, a_2, and a_3 are the semi-axes and $theta_1$, $theta_2$, and $theta_3$ are the rotational angles around the x, y, and z-axes in radians. If $a_1 = a_2 = a_3$ the ellipsoid is a ball. The parameters for the 3-D Gaussian function are: $[2, mu_1, mu_2, mu_3, sigma_1, sigma_2, sigma_3, cor_{xy}, cor_{xz}, cor_{yz}]$, where (mu_1, mu_2, mu_3) is the center, $sigma_1$, $sigma_2$, and $sigma_3$, are the standard deviations, and cor_{xy}, cor_{xz}, and cor_{yz} are the correlations. The parameters for the 3-D box are: $[3, x_0, y_0, z_0, a_1, a_2, a_3]$, where (x_0, y_0, z_0) is the center and a_1, a_2, and a_3 are half the lengths of the sides of the box.
rho	The material parameter of the objects. The parameter *rho* can either be given as a scalar, where all objects have the same material value or as a vector, where material values are assigned from left to right and top to bottom. Default material value is 1.
gausstol	The tolerance for the material parameter of the Gaussian object before cut off. The default is set to 0.

Table 2.1

The Matlab function `geophantoms` creates a matrix $\mathbf{X}$ of the dimensions specified in $\mathbf{N}$. This phantom can either be one of the predefined examples or a user customized phantom. The phantom can be used when creating numerical geological test problems. This function is used in the test problems `gravity2D`, `gravity3D`, `fresneltomo2D`, and `fresneltomo3D`.

2-D	**3-D**
Disk	Ball
Ellipse	Ellipsoid
Filled box	Filled box
2-D Gaussian function	3-D Gaussian function

Table 2.2: Implemented shapes in `geophantoms`.

A phantoms is in general a subsurface area that have a material value of zero except in some specific areas. These areas are implemented as geometrical shapes. In `geophantoms` several geometrical shapes has been implemented, see Table 2.2. Furthermore, four predefined examples are implemented, these examples consist of one or two kinds of shapes. In 2-D the subsurface area has the length 1 in arbitrary units in the x-direction and the depth in the z-direction is d. This depth will usually be less than 1. The z-axis points downwards with 0 at the surface, making it a left-hand coordinate system. In 3-D the subsurface has the length 1 in arbitrary units in the x-direction, the width in the y-direction is d_y and the depth z-direction is d_z. Both d_y and d_z will usually be less than 1. Again, the z-axis points downwards with 0 at the surface, making it a left-hand coordinate system. The four examples implemented in two and three dimensions are described in Table 2.3.

Ex.	**2-D**	**3-D**
1	$f(x, z)$ is two disks of different sizes. One with center at (0.4, $0.2d$) with radius $0.07d$ and the other with center at (0.8, $0.4d$) and radius $0.15d$.	$f(x, y, z)$ is two balls of different sizes. One with center at (0.4, $0.4d_y$, $0.2d_z$) and radius $0.07{\cdot}\min(d_y, d_z)$ and the other with center at $(0.8, 0.75d_y, 0.4d_z)$ with radius $0.15{\cdot}\min(d_y, d_z)$.
2	$f(x, z)$ is two ellipses and a disk. The disk has its center at (0.25, $0.15d$) and radius $0.05d$. The first ellipse has its center at (0.25, $0.35d$) with $a_1 = 0.07d$ and $a_2 = 0.2d$ and it is rotated $-\pi/4$. The other ellipse has its center at (0.75, $0.5d$) with a_1 being $0.07d$ and a_2 being $0.2d$ and is rotated $\pi/3$.	$f(x, y, z)$ is two ellipsoids and a ball. The ball has its center at (0.25, $0.25d_y$, $0.15d_z$) and a radius of $0.1{\cdot}\min(d_y, d_z)$. The first ellipsoid has its center at (0.25, $0.50d_y$, $0.35d_z$), a_1 and a_2 are $0.07{\cdot}\min(d_y, d_z)$ in x-direction and y-direction, and a_3 is $0.2{\cdot}\min(d_y, d_z)$ in z-direction. It is rotated $-\pi/4$ around the y-axis. The other ellipsoid has center at (0.75, $0.75d_y$, $0.5d_z$), a_1 and a_2 are $0.1{\cdot}\min(d_y, d_z)$ in x-direction and y-direction, and a_3 is $0.25{\cdot}\min(d_y, d_z)$ in the z-direction. It is rotated $\pi/3$ around the y-axis.

3	$f(x, z)$ is a Gaussian object and a small disk. The disk has its center at $(0.3,\ 0.2d)$ and a radius of $0.05d$. The Gaussian object has its center at $(0.4,\ 0.4d)$ and has standard deviations of $[0.4,\ 0.1]\cdot d$. The correlation between x and z is 0.01.	$f(x, y, z)$ is a Gaussian object and a small ball. The ball has its center at $(0.3,\ 0.75d_y,\ 0.2d_z)$ and a radius of $0.1\cdot\min(d_y,\ d_z)$. The Gaussian object has its center at $(0.4,\ 0.2d_y,\ 0.4d_z)$ and has standard deviations of $[0.4,\ 0.1,\ 0.1]\cdot\min(d_y,\ d_z)$. The correlations r_{xy}, r_{xz}, and r_{yz} are respectively 0.7, 0.0, and 0.6.
4	$f(x, z)$ is two long boxes that overlap vertically. The first box has its center at $(0.35,\ 0.2d)$ and half lengths of $[0.25,\ 0.05d]$. The second box has its center at $(0.65,\ 0.4d)$ and half lengths of $[0.25,\ 0.05d]$. This example is suited for problems with small depth, i.e. $N_x \gg N_z$.	$f(x, y, z)$ is two flat boxes that overlap vertically. The first box has its center at $(0.35,\ 0.35d_y,\ 0.2d_z)$ and half lengths of $[0.25,\ 0.25d_y,\ 0.05d_z]$. The second box has its center at $(0.65,\ 0.65d_y,\ 0.4d_z)$ and half lengths of $[0.25,\ 0.25d_y,\ 0.05d_z]$. This example is suited for problems with small depth, i.e. $N_x \gg N_z$.

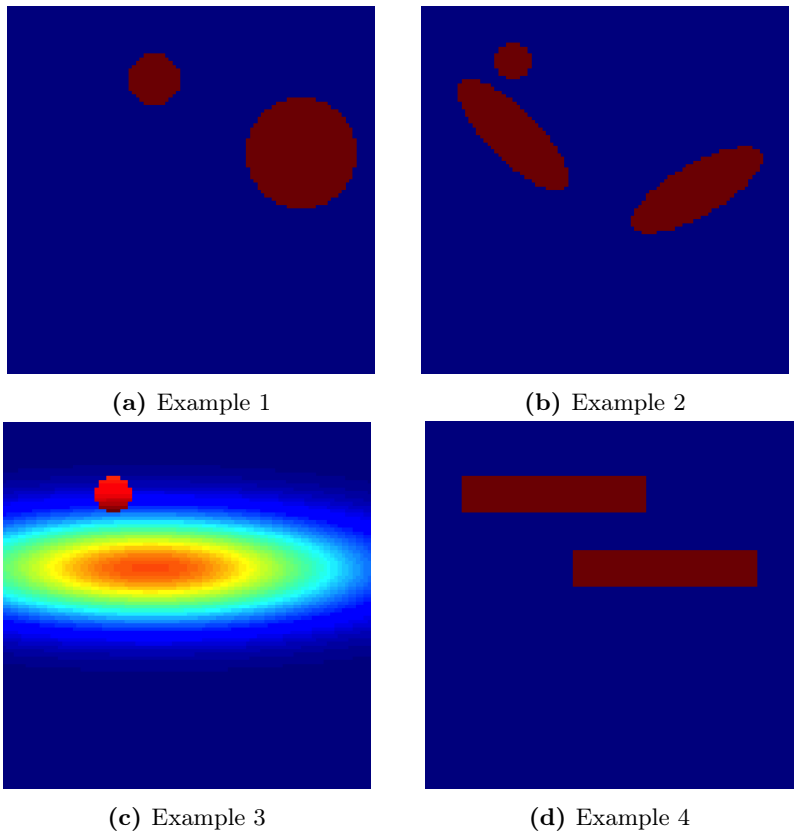
Table 2.3: Implemented examples in `geophantoms`.

If the implemented examples are not sufficient for the user, then the user can specify their own phantom by creating a geometric shape with the option field *parameters*. To create a phantom consisting of more than one shape, one must create a sum of individually created shapes, that is for example, $X = X_{ellipse} + X_{box}$.

When using the option *parameters*, the box is implemented so the user specifies the center by (x_0, y_0, z_0) and half the lengths of the box, a_1, a_2, and a_3. The box has the dimensions of $2 \cdot a_1 \times 2 \cdot a_2 \times 2 \cdot a_3$. It is assigned the material value *rho*. In the 2-D version only (x_0, z_0) and two half lengths are needed.

An ellipse is implemented with center, c, in (x_0, z_0), semi-axes a_1 and a_2. The parameter θ is the rotational angle in radians. It is a standard ellipse which is rotated with a rotation matrix [10],

$$\begin{aligned} &\frac{1}{a_1^2}\big((x - x_0)\cos(\theta) + (z - z_0)\sin(\theta)\big)^2 \\ +&\frac{1}{a_2^2}\big(-(x - x_0)\sin(\theta) + (z - z_0)\cos(\theta)\big)^2 \leq 1. \end{aligned} \tag{2.4}$$



(a) Example 1 **(b)** Example 2

(c) Example 3 **(d)** Example 4

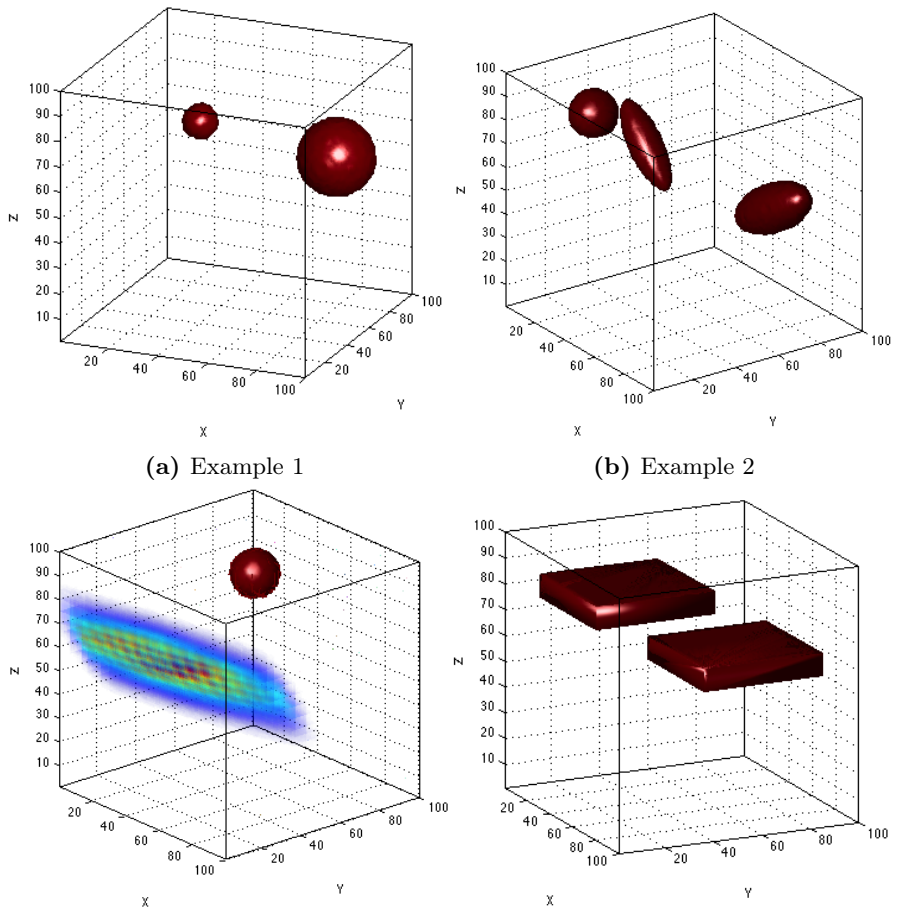
Figure 2.2: The four predefined examples in 2-D.

Every pixel with center (x, z) satisfying equation (2.4), is then given the material value *rho*. The disk is just an ellipse with $a_1 = a_2$. The ellipse in 3-D is the ellipsoid and can be found in appendix A. The ball in 3-D is the ellipsoid where all the semi-axes have the same length.

The 2-D Gaussian function [9] for each point (x, z) is implemented as,

$$f(x,z) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-r^2}} \cdot \exp\left(\frac{-1}{2(1-r^2)}\left(\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(z-\mu_2)^2}{\sigma_2^2} - \frac{2r(x-\mu_1)(z-\mu_2)}{\sigma_1\sigma_2}\right)\right),$$

with $\sigma = [\sigma_1, \sigma_2]$ being the variance respectively in the x and z-directions. The variable r is the correlation between x and z, $\mu = [\mu_1, \mu_2]$ are the means and the center of the object. The Gaussian function is assigned the material value *rho*



(a) Example 1 **(b)** Example 2

(c) Example 3 **(d)** Example 4

Figure 2.3: The four predefined examples in 3-D.

at the center and it then decays outwards. As for the ellipsoid, the Gaussian 3-D function can be found in appendix A.

Examples:

The 2-D default example, number 1, is set up with a phantom of size 100×100:

```
N = [100 100];
X = geophantoms(N);
```

and is visualized in Figure 2.2a with:

```
imagesc(X)
axis image off
```

Example 2 is chosen and visualized the same way as before in Figure 2.2b:

```
example = 2;
X = geophantoms(N,example);
```

Example 3 and 4 in 2-D are displayed respectively in Figures 2.2c and 2.2d. The same is done for the 3-D examples, just with a phantom of size $100 \times 100 \times 100$ and are displayed in Figure 2.3, they are visualized with the Matlab function Sliceomatic [6]. For examples of the use of Sliceomatic see `gravity3Ddemo` (script).
A user defined phantom can be created using the field *parameters* in the struct *options*. Here a disk with center in $(0.3, 0.5)$ and radius 0.3 and a box with center in $(0.6, 0.5)$ and half lengths of 0.3 in the x-direction and 0.1 in the z-direction. They are then added together, such that the overlap between the shapes has the sum of the material values, and visualized in Figure 2.4.

```
options.parameters = [1,0.3,0.5,0.3,0.3,0];
X1 = geophantoms(N,[],options);
options.parameters = [3,0.6,0.5,0.3,0.1];
X2 = geophantoms(N,[],options);
X = X1 + X2;
imagesc(X)
axis image off
```

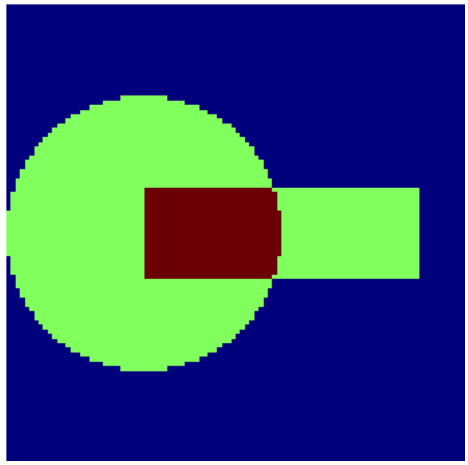


Figure 2.4: The user defiend phantom.

For use of phantoms in test problems see sections: 2.2, 2.3, 3.1, and 3.2.

See also:

`gravity2D, gravity3D, fresneltomo2D,`, `fresneltomo3D`, `gravity2Ddemo` (script), `gravity3Ddemo` (script), `tomo2Ddemo` (script), `tomo3Ddemo` (script)

2.2 gravity2D

Purpose:

Test problem: 2-D gravity surveying model problem.

Synopsis:

```
[A,b,x,sx] = gravity2D(N,m)
[A,b,x,sx] = gravity2D(N,m,options)
```

Description:

Input Parameters	
N	An integer or a vector of length 2 that defines the size of the subsurface area, measured in pixels. If N is given as a scalar, the area becomes quadratic, $N = N_x = N_z$. When N is given as a vector then $N = [N_x\ N_z]$.
m	The number of measurement points on the surface.
options	Struct containing the option fields for the test problem, see below.

Output Parameters	
A	The system matrix of size $m \times n$, where $n = N_x \cdot N_z$.
b	The right hand-side as a vector of length m.
x	The phantom $\mathbf{X}$ as a vector of length $n = N_x \cdot N_z$. The columns of the phantom are stacked by vec($\mathbf{x}$) and is reshaped by `X=reshape(x,Nz,Nx)`.
sx	A vector of length m with the x-coordinates of measurement points.

Option Fields	
groups	The number of groups the measurement points should be divided into. See Figure 2.5.
mspace	The number of skipped potential measuring points in an equidistant grid. The default value is set to 2, and is only applicable when the number of groups is greater than 1. See Figure 2.5.
offset	A symmetric extension of the measuring interval. The parameter *offset* is the length which is added to both sides of the interval. It can also be negative, in which case the measuring interval is shortened. Default value is set to 0. See Figure 2.5.

example	A number specifying the phantom example. Valid example numbers are 1 to 4, see section 2.1.
rho	The mass density of the objects. The parameter *rho* can either be given as a scalar, where all objects have the same mass density or as a vector, where densities are assigned from left to right and top to bottom. Default density is 1.
gausstol	The tolerance for the mass density of the Gaussian object before cut off. The default is set to 0.

Table 2.4

The Matlab function `gravity2D` creates a mass density distribution, $\mathbf{x}$, the system matrix, $\mathbf{A}$, and the gravity measurements, $\mathbf{b}$. It furthermore gives the coordinates of the measurement points in $\mathbf{sx}$. The mass distribution is created using the function `geophantoms`, see section 2.1.

The problem is modeled by a first kind Fredholm integral with the kernel,

$$K(s_x, x, z) = \frac{z}{(z^2 + (s_x - x)^2)^{3/2}},$$

and it is discretized using the midpoint method, see the beginning of section 2. This results in the elements of $\mathbf{A}$ being of the form,

$$a_{i,j} = \frac{d}{n} \frac{z_k}{(z_k^2 + (s_x - x_l)^2)^{3/2}},$$

where $n = N_x \cdot N_z$, $j = k + (l-1)N_z$, and,

$$x_l = \frac{l - \frac{1}{2}}{N_x}, \quad z_k = \frac{(k - \frac{1}{2})d}{N_z}, \quad l = 1...N_x, k = 1...N_z.$$

The function has several options regarding the measurement points, see Table 2.4 and Figure 2.5. The first option field is *groups*, which specifies the number of groups that the measurement points are divided into. Each group has the same number of measurement points if possible, otherwise the last groups, i.e. the ones with the highest x-coordinates, have one more point than the first groups. The space between the groups are defined using *mspace*. The option *mspace* defines the number of measurement points there could be in the gap, if these potential measurement points also were equidistant. The interval, in which the

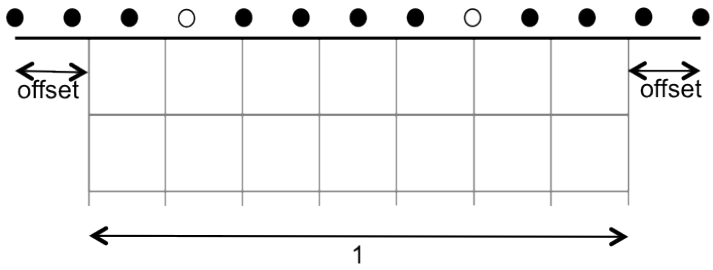


Figure 2.5: Illustration of the options. Here $m = 11$, *groups* $= 3$ and *mspace* $=$ 1. Notice that the last groups have one more point.

measurements are taken, can be extended using *offset*. The option *offset* gives the length that the measurement interval will be extended with to each side. The measurement interval is as default set to $[0, 1]$ and a positive offset will make it larger and a negative smaller, that is, $[0-\mathit{offset}, 1+\mathit{offset}]$. All of these options are illustrated in Figure 2.5.

The rest of the options are regarding the phantom, that is, *example*, *rho*, and *gausstol*. For the use of these, see the function `geophantoms` in section 2.1.

Examples:

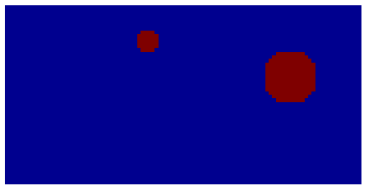
The problem is set up with a subsurface area of size 100×50 and 100 measurement points.

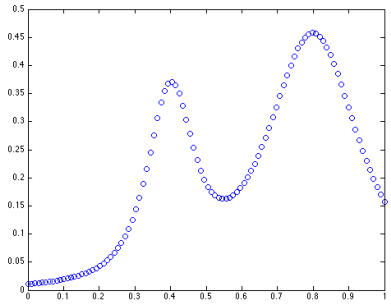
```
N = [100 50];
m = 100;
[A,b,x,sx] = gravity2D(N,m);
```

The phantom, $\mathbf{x}$, and the right-hand side, $\mathbf{b}$, of the problem can be visualized with the following code:

```
imagesc(reshape(x,N(2),N(1)))
axis image off
figure
plot(sx,b,'o');
```

and are displayed in Figure 2.6.





(a) Exact phantom, phantom example 1 **(b)** Right-hand side

Figure 2.6

The number of measurement groups are changed by changing the option field *groups*,

```
options.groups = 3;
[A,b,x,sx] = gravity2D(N,m,options);
```

The other fields in *options* can be changed and applied in the same way.

See also:

`geophantoms`, `gravity3D`, `gravity2Ddemo` (script)

References:

Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*, SIAM, 2010.

2.3 gravity3D

Purpose:

Test problem: 3-D gravity surveying model problem.

Synopsis:

```
[A,b,x,sxy] = gravity3D(N,m)
[A,b,x,sxy] = gravity3D(N,m,options)
```

Description:

Input Parameters	
N	An integer or a vector of length 3 that defines the size of the subsurface volume. If N is given as a scalar, the volume becomes cubic, $N = N_x = N_y = N_z$. When N is given as a vector then $N = [N_x\ N_y\ N_z]$.
m	An integer or a vector of length 2 that defines the number of measurement points in each direction. If m is an integer then $m = m_x = m_y$ and if it is given as a vector then $m = [m_x\ m_y]$.
options	Struct containing the option fields for the test problem, see below.

Output Parameters	
A	The system matrix of size $(m_x \cdot m_y) \times n$, where $n = N_x \cdot N_y \cdot N_z$.
b	The right-hand side vector of length $m_x \cdot m_y$.
x	The phantom $\mathbf{X}$ as a vector of length $n = N_x \cdot N_y \cdot N_z$. The columns of the phantom are stacked with $\mathbf{x} = \text{vec}(\mathbf{X})$ and is reshaped by `X=reshape(x,Ny,Nx,Nz)`.
sxy	A matrix of size $2 \times (m_x \cdot m_z)$ with **sxy**$= [s_x, s_y]$, which are the coordinates of measurement points.

Option Fields	
groups	An integer or a vector of length 2 that defines the number of groups the measurement points should be divided into. If *groups* is a vector, the first element defines the number of groups in the x-direction and the second element in the y-direction. If it is an integer there will be the same number of groups in each direction. The default value is 1. See Figure 2.7.

mspace	An integer or a vector of length 2. It defines the number of skipped potential measuring points in an equidistant grid. If it is a vector the first element defines the number of skipped points in the x-direction and the second element in the y-direction. The default value is set to 2, and is only applicable when the number of groups is greater than 1. See Figure 2.7.
offset	A symmetric extension of the measuring interval. The parameter *offset* is the length which is added to all four sides of the interval. It can also be negative, in which case the measuring intervals are shortened. Default value is set to 0. See Figure 2.7.
example	A number specifying the phantom example. Valid example numbers are 1 to 4, see section 2.1.
rho	The mass density of the objects. The parameter *rho* can either be given as a scalar, where all objects have the same mass density or as a vector, where densities are assigned from left to right and top to bottom. Default density is 1.
gausstol	The tolerance for the mass density of the Gaussian object before cut off. The default is set to 0.

Table 2.5

The Matlab function `gravity3D` creates a mass density distribution, $\mathbf{x}$, a system matrix, $\mathbf{A}$, and the gravity measurements, $\mathbf{b}$. It also gives the coordinates of the measurement points out as a matrix, $\mathbf{sxy}$. The mass density distribution is constructed using `geophantoms`, see section 2.1.

The problem is modeled by a first kind Fredholm integral with the kernel,

$$K(s_x, s_y, x, y, z) = \frac{z}{(z^2 + (s_y - y)^2 + (s_x - x)^2)^{3/2}},$$

and it is discretized using the midpoint method [8]. This results in the entries of $\mathbf{A}$ being of the form,

$$a_{i,j} = \frac{d_y \cdot d_z}{n} \frac{z_{k_3}}{(z_{k_3}^2 + (s_y - y_{k_2})^2 + (s_x - x_{k_1})^2)^{3/2}},$$

where $n = N_x \cdot N_y \cdot N_x$, $j = k_1 + (k_2 - 1)N_x + (k_3 - 1)N_x N_y$, and,

$$x_{k_1} = \frac{k_1 - \frac{1}{2}}{N_x}, \quad y_{k_2} = \frac{(k_2 - \frac{1}{2})d_y}{N_y}, \quad z_{k_3} = \frac{(k_3 - \frac{1}{2})d_z}{N_z},$$

where $k_1 = 1...N_x, k_2 = 1...N_y, k_3 = 1...N_z$.

The function `gravity3D` has several options regarding how to place the measurement points, see Table 2.5 and Figure 2.7.

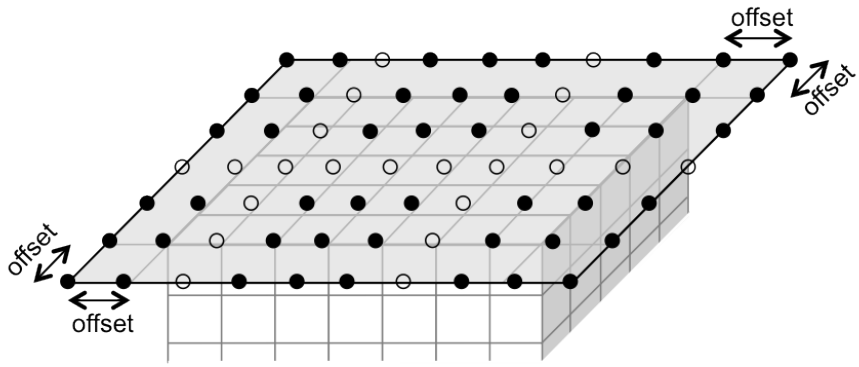


Figure 2.7: Illustration of the options. Here $m_x = 8$, $m_y = 6$, *groups* $= [3, 2]$ and *mspace* $= 1$. Notice that the last two groups have one more point in each line in the x-direction.

The first option field is *groups*, which specifies the number of groups the measurement points are divided into both in the x-direction and the y-direction. Each group in the x-direction has the same number of points if possible. If this is not possible, the last groups, i.e. the ones with the highest x-coordinates, will have one more row of points than the firsts. This is the same for the y-direction, where the last groups have the highest y-coordinates.
The option *mspace* defines the space between the groups. That is, the number of measurement points there could be in the gap, if these potential measurement points also were equidistantly placed in both dimensions. The interval, in which the measurements are taken, can be extended using *offset*. The option *offset* gives the length that the measurement interval will be extended with in both ends of the x and y-intervals. The measurement interval is as default set to $[0, 1] \times [0, d_y]$, and a positive offset will make it larger, and a negative smaller, that is, $[0 - \mathit{offset}, 1 + \mathit{offset}] \times [0 - \mathit{offset}, d_y + \mathit{offset}]$, where $d_y = N_y/N_x$. All of these options are illustrated in Figure 2.7.

The rest of the options are regarding the phantom, that is, *example*, *rho*, and *gausstol*. For the use of these, see the function `geophantoms` in section 2.1.

Examples:

The problem is set up with subsurface volume of size $35 \times 25 \times 15$ and 30 measurement points in the x-direction and 20 y-direction.

```
N = [35 25 15];
m = [30 20];
[A,b,x,sxy] = gravity3D(N,m);
```

Since our coordinate system has 0 at the surface the z-axis is flipped in the illustration. The phantom, $\mathbf{x}$, is reshaped and flipped to be visualized in Figure 2.8a.

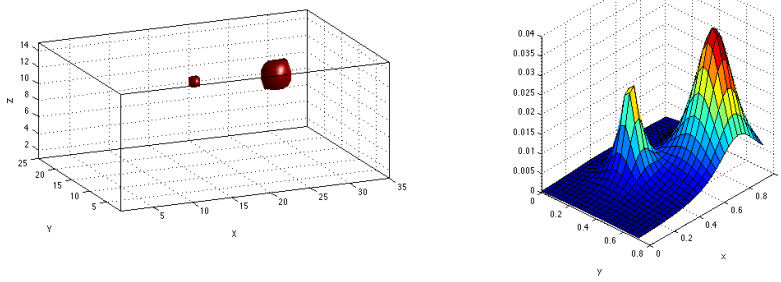
```
sliceomatic(flipdim(reshape(x,N(2),N(1),N(3)),3))
```

We use the Matlab Package Sliceomatic for 3-D visualizations [6].

The right-hand side, $\mathbf{b}$, of the problem can be visualized with the following code

```
surf(reshape(sxy(:,1),m(2),m(1)),reshape(sxy(:,2),...
 m(2),m(1)),reshape(b,m(2),m(1)));
```

and can be seen in Figure 2.8b.



(a) Exact phantom, phantom example 1 **(b)** Right-hand side

Figure 2.8

The field *offset* is activated and set by,

```
options.offset = 0.2;
[A,b,x,sxy] = gravity3D(N,m,options);
```

The other fields in *options* can be changed and applied in the same way.

See also:

`geophantoms`, `gravity2D`, `gravity3Ddemo` (script)

References:

Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*, SIAM, 2010.

Chapter 3

Seismic Tomography

To perform tomography is to create images of sections of the insides of an object by using waves to penetrate it. In this section, we will create test problems, where seismic waves are used to reconstruct an image of the subsurface of the earth. We consider a setup of sources and receivers. The sources send out seismic waves and the receivers are seismographs, which detect the waves arriving at their positions and measure the travel time of the waves. An example of a setup can be seen in Figure 3.1. Here, the sources are marked as stars, and have coordinates (s_x, s_z), and the receivers with crosses, and have coordinates (p_x, p_z). The coordinate system has $(0, 0)$ at the upper left corner of the subsurface area with x-axis to the right and and the z-axis downwards, making it a left-hand coordinate system. The sources and half of the receivers are located in boreholes and the rest of the receivers on the surface.

When modeling the travel time of a wave from one source to a receiver, a simple model of the path being a ray is often used as in section 7.7 in [5]. The model can be derived as follows. We let $f(x, z)$ be a material parameter describing how much the material slows down the wave, and $d\tau$ be an infinitesimal small part of the ray at a point (x, z). Then the travel time of the wave is $f(x, z)d\tau$. The travel time along an entire ray i between one pair of sources and receivers

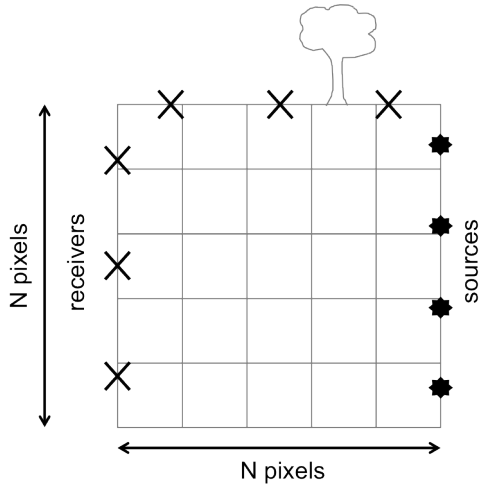


Figure 3.1: Geometry of a two dimensional seismic tomography problem.

are,

$$b_i = \int_{\text{ray } i} f(x,z)d\tau, \tag{3.1}$$

where $(x,z) = (x_0,z_0) + \tau \mathbf{d}_i$. The sources have the positions (x_0,z_0), and $\mathbf{d}_i$ is a unit vector in the ray direction. If we discretize this problem into $N \times N$ pixels, as shown in Figure 3.1, and let $f(x,z) = f_{k,l}$ be constant in each pixel (k,l), then this gives the travel time for each ray i,

$$b_i = \sum_{(k,l)\in \text{ray } i} f_{k,l}\Delta\tau_{k,l}^{(i)}, \tag{3.2}$$

where $\Delta\tau_{k,l}^{(i)}$ is the length of ray i in pixel (k,l).

In these test problems, we will model the area in which the wave travels by a wider shape than a ray. This is done since it is a more accurate approximation to the natural seismic wave according to [11]. This path is defined in two

dimensions as the following Fresnel kernel [7],

$$K(\Delta t, w, \alpha) = \cos\left(\frac{2\pi\Delta t}{w}\right)\exp\left(-\left(\frac{\alpha\Delta t}{w/4}\right)^2\right), \tag{3.3}$$

where

$$\begin{aligned}\Delta t = \frac{1}{N}\Big(&\sqrt{(s_x-x)^2+(s_z-z)^2}\\ &+\sqrt{(p_x-x)^2+(p_z-z)^2}\\ &-\sqrt{(s_x-p_x)^2+(s_z-p_z)^2}\Big).\end{aligned} \tag{3.4}$$

The square roots in (3.4) are divided by N, so that the width of the kernel does not depend on the discretization. The parameter w determines the width of the kernel in arbitrary units together with the parameter α that determines the exponential decay. A high value of w gives a wide kernel, and a high value of α gives a high exponential decay, i.e. a more narrow kernel. The coordinates x and z are the coordinates for each center of the pixels (k, l). An example of the shape of this wave path with width $w = 0.2$ and $\alpha = 70$ can be seen in Figure 3.2.

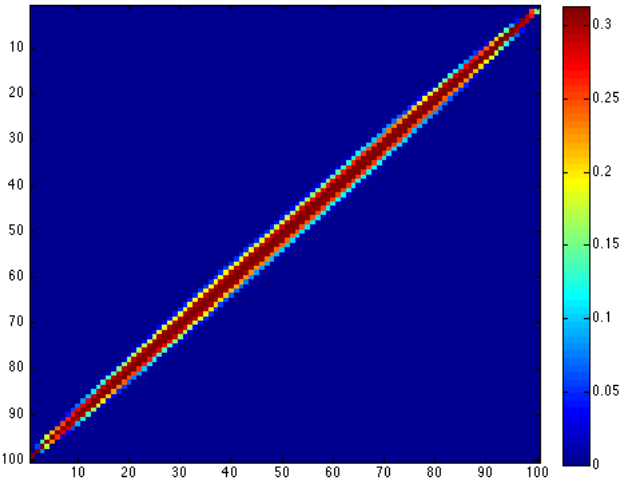


Figure 3.2: The Fresnel kernel of the 2-D problem.

We calculate the wave path for one pair of sources and receivers for all the pixels in the subsurface. This corresponds to sending one wave from a source to

a receiver. The value assigned to each pixel in the subsurface matrix is a weight of how much of the wave that passes through the pixel, this is corresponding to $\Delta\tau_{k,l}^{(i)}$ in the simple ray model in equation (3.2). The pixel matrix, as the one in Figure 3.2, is then vectorized to be a row in the system matrix $\mathbf{A}$. When we have gone through all possible pairs of sources and receivers we have the entire system matrix of size $(s \cdot p) \times N^2$ in 2-D and $(s \cdot p) \times N^3$ in 3-D.

The input, $\mathbf{x}$, of the inverse system is a geological phantom created with `geophantoms`, see sections 2.1. It represents the slowness of the subsurface in each pixel. That is, how much it slows down the wave when it penetrates the pixel and it is corresponding to $f(x, z) = f_{k,l}$ in equation (3.2). The output, $\mathbf{b}$, of the system (1.1) is found by the matrix-vector multiplication, $\mathbf{Ax} = \mathbf{b}$. It represents the travel time between each source-receiver pair, so each element corresponds to a b_i in equation (3.2).

3.1 fresneltomo2D

Purpose:

Test problem: 2-D seismic tomography problem.

Synopsis:

```
[A,b,x,sxz,pxz] = fresneltomo2D(N)
[A,b,x,sxz,pxz] = fresneltomo2D(N,s)
[A,b,x,sxz,pxz] = fresneltomo2D(N,s,p)
[A,b,x,sxz,pxz] = fresneltomo2D(N,s,p,options)
```

Description:

Input Parameters	
N	An integer specifying the size of the square subsurface grid. The grid will be of size $N \times N$.
s	An integer specifying number of source points. The default value is N.
p	An integer specifying number of receiver points. The default value is N.
options	Struct containing the option fields for the test problem, see below.

Output Parameters	
A	The system matrix of size $(s \cdot p) \times N^2$.
b	The right-hand side column vector of length $s \cdot p$.
x	The phantom $\mathbf{X}$ as a column vector of length N^2, where the columns are stacked with $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, i.e. `X = reshape(x,N,N)`.
sxz	A matrix with coordinates of the source points. The x-coordinates are in the first column and z-coordinates in second column.
pxz	A matrix with coordinates of the receiver points. The x-coordinates are in the first column and z-coordinates in second column.

Option Fields	
width	Specifies the width of the Fresnel kernel in arbitrary units. Should be a positive scalar and we recommend values between 0 and 1. Default is 0.2.

alpha	Positive exponential decay parameter as a scalar. It influences the width of the kernel. We recommend values higher than 20. Default is 70.
kerneltol	Tolerance on the kernel before cut off. Default is 10^{-6}.
example	Number that specifies the phantom example. Valid example numbers are 1 to 4, see `geophantoms` in section 2.1 for details. Default is 1.
rho	Parameter describing the slowness of objects, can be a scalar or a vector. If *rho* is a vector, the different slowness parameters are assigned to the objects from left to right and top down. Default is 1.
gausstol	Tolerance on slowness for Gaussian objects before cut off. Default is 0.

Table 3.1

The Matlab function `fresneltomo2D` creates a representation of a subsurface area with a phantom stacked in a vector, $\mathbf{x}$, with length N^2, the system matrix, $\mathbf{A}$, of the size $s \cdot p \times N^2$, and the right-hand side, $\mathbf{b}$, with length $s \cdot p$. The test problem is used to model how a wave travels from a source point, s, to a receiver, p, through areas of different slowness distributions. The subsurface is represented by a $N \times N$ grid of pixels, where the sources are equidistantly placed along the right edge and the receivers are equidistantly placed along the left edge and on the surface. The geological phantom is placed in the grid, and the possible phantoms are described in sections 2.1.

If the Matlab function is called without specifying the number of sources, s, and receivers, p, then they will respectively be set to N. The used phantom is specified in *example* in options, see Table 3.1. If no phantom is specified, then a phantom of two disks of different sizes is used, which is example 1, see section 2.1.

In this implementation the 2-D wave path from one source to one receiver is modeled by a shape wider than a ray. This shape is made with the kernel [7]

$$K(\Delta t, w, \alpha) = \cos\left(\frac{2\pi\Delta t}{w}\right)\exp\left(-\left(\frac{\alpha\Delta t}{w/4}\right)^2\right), \tag{3.5}$$

where

$$\Delta t = \frac{1}{N}\Big(\sqrt{(s_x - x)^2 + (s_z - z)^2} + \sqrt{(p_x - x)^2 + (p_z - z)^2} - \sqrt{(s_x - p_x)^2 + (s_z - p_z)^2}\Big). \tag{3.6}$$

The square roots in equation (3.6) are divided by N, so that the width of the kernel does not depend on the discretization. The parameter w is the width of the kernel in arbitrary units and is called *width* in Matlab. The parameter α influences the exponential decay and is called *alpha* in Matlab. A high α gives a high decay, i.e. a more narrow kernel. The coordinates x and z are the coordinates for each center of the pixels. The default values of *width* and *alpha* are chosen because they give a sparsity of the matrix $\mathbf{A}$ of about 5%. To see how the matrix $\mathbf{A}$ is constructed see section 3.

Examples:

The problem is set up with a subsurface area of size 100×100, that is, $N = 100$, and 10 sources and 15 receivers.

```
N = 100;
s = 10;
p = 15;
[A,b,x,sxz,pxz] = fresneltomo2D(N,s,p);
```

The phantom, $\mathbf{x}$, and the sources and receivers can be visualized with the following code:

```
imagesc(reshape(x,N,N))
axis('image',[0 100 0 100],'off')
hold on
plot(sxz(:,1),sxz(:,2),'g*','MarkerSize',20,'linewidth',2);
hold on
plot(pxz(:,1),pxz(:,2),'mx','MarkerSize',20,'linewidth',2);
```

and are displayed in Figure 3.3.

The exponential decay parameter *alpha* is changed by

```
options.alpha = 30;
[A,b,x,sxz,pxz] = fresneltomo2D(N,s,p,options);
```

The other fields in *options* can be changed and applied in the same way.

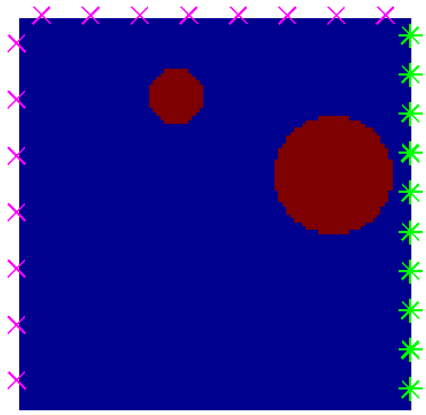


Figure 3.3: Exact phantom with sources (*) and receivers (x), phantom example 1.

See also:

`geophantoms`, `fresneltomo3D`, `tomo2Ddemo` (script)

Limitations:

If *width* is set too high, above 1, and *alpha* is set too low, below 20, then **A** will not be sparse and the function will be slow.

References:

Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*, SIAM, 2010.

Jensen, J. M., B. H. Jacobsen, and J. Christensen-Dalsgaard, *Sensitivity kernels for Time-Distance inversion*, Solar Phys.: SOHO9 topical issue (2000), pp. 231-239

3.2 fresneltomo3D

Purpose:

Test problem: 3-D seismic tomography problem.

Synopsis:

```
[A,b,x,sxyz,pxyz] = fresneltomo3D(N)
[A,b,x,sxyz,pxyz] = fresneltomo3D(N,s)
[A,b,x,sxyz,pxyz] = fresneltomo3D(N,s,p)
[A,b,x,sxyz,pxyz] = fresneltomo3D(N,s,p,options)
```

Description:

Input Parameters	
N	An integer specifying the size of the cubic subsurface grid. The grid will be of size $N \times N \times N$.
s	An integer specifying number of source points. The default value is N.
p	An integer specifying number of receiver points. The default value is N.
options	Struct containing the option fields for the test problem, see below.

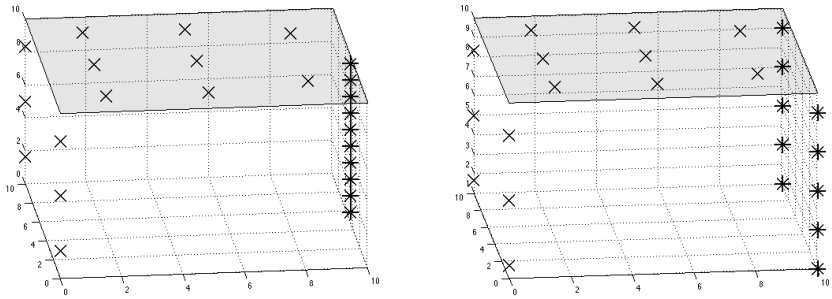
Output Parameters	
A	The system matrix of size $(s \cdot p) \times N^3$.
b	The right-hand side column vector of length $s \cdot p$.
x	The phantom $\mathbf{X}$ as a column vector of length N^3, where the columns are stacked with $\mathbf{x} = \text{vec}(\mathbf{X})$, i.e. `X = reshape(x,N,N,N)`.
sxyz	A matrix with coordinates of the source points. The x-coordinates are in the first column, y-coordinates in the second column, and z-coordinates in third column.
pxyz	A matrix with coordinates of the receiver points. The x-coordinates are in the first column, y-coordinates in the second column, and z-coordinates in third column.

Option Fields	
width	Specifies the width of the Fresnel kernel in arbitrary units. Should be a positive scalar and we recommend values between 0 and 1. Default is 0.5.
alpha	Positive exponential decay parameter as a scalar. It influences the width of the kernel. We recommend values higher than 2. Default is 10.
sourceHoles	The number of boreholes, can either be 1 or 2. The default is 1.
kernel	The kernel type used to construct $\mathbf{A}$. Can either be solid or hollow. The default kernel type is solid.
kerneltol	Tolerance on the kernel before cut off. Default is 10^{-6}.
example	Number that specifies the phantom example. Valid example numbers are 1 to 4, see `geophantoms` in section 2.1 for details. Default is 1.
rho	Parameter describing the slowness of objects, can be a scalar or a vector. If *rho* is a vector, the different slowness parameters are assigned to the objects from left to right and top down. Default is 1.
gausstol	Tolerance on slowness for Gaussian objects before cut off. Default is 0.

Table 3.2

The Matlab function `fresneltomo3D` creates a representation of a subsurface volume with a geophantom stacked in a vector, $\mathbf{x}$, with length N^3, the system matrix, $\mathbf{A}$, of the size $(s{\cdot}p)\times N^3$, and the right-hand side, $\mathbf{b}$, with length $s{\cdot}p$. The test problem is used to model how a wave travels from a source point to a receiver through areas of different slowness distributions. The subsurface is represented by a $N \times N \times N$ grid of voxels, where the sources are equidistantly placed and send waves through the subsurface to the equidistantly placed receivers. If the option *sourceHoles* is set to 1, the sources are placed in a column at $(N, N/2, \mathbf{z})$ as in Figure 3.4a. If *sourceHoles* is set to 2, the sources are placed in two columns at $(N, 0, \mathbf{z})$ and $(N, N, \mathbf{z})$ as in Figure 3.4b. The receivers are always placed in two columns at $(0, 0, \mathbf{z})$ and $(0, N, \mathbf{z})$ each with $\frac{1}{6}p$. The remaining $\frac{2}{3}p$ is placed in a grid-like pattern on the surface, see Figure 3.4. The phantom is placed in the voxel grid, and possible phantoms are described in sections 2.1.

If the Matlab function is called without specifying the number of sources, s, and receivers, p, then they will respectively be set to N. The used phantom is specified in *example* in options, see Table 3.2. If no phantom is specified, then a phantom of two balls of different sizes is used, which is example 1, see section



(a) One source hole **(b)** Two source holes

Figure 3.4: The placement of the 10 sources and 14 receivers.

2.1.

In this implementation, the 3-D wave path from one source to one receiver is modeled by a shape wider than a ray. This shape can be modeled by two different kind of Fresnel kernels. The option *kernel* specifies if the kernel is solid or hollow.
The solid kernel is of the following form [7]

$$K(\Delta t, w, \alpha) = \cos\left(\frac{2\pi\Delta t}{w}\right)\exp\left(-\left(\frac{\alpha\Delta t}{w/4}\right)^2\right), \tag{3.7}$$

and is a direct expansion of the one used in 2-D, see section 3. An example of the kernel can be seen in Figure 3.5a. The hollow kernel is of the following form [7]

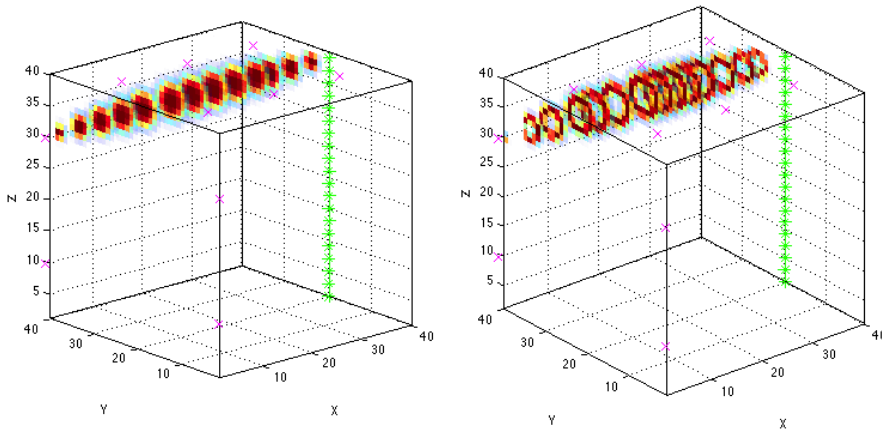
$$K(\Delta t, w, \alpha) = \sin\left(\frac{4\pi\Delta t}{w}\right)\exp\left(-\left(\frac{\alpha\Delta t}{w/4}\right)^2\right), \tag{3.8}$$

and is a more accurate model of the actual 3-D wave path and an example can be seen in Figure 3.5b. In both cases

$$\begin{aligned}\Delta t = \frac{1}{N}\Big(&\sqrt{(s_x-x)^2+(s_y-y)^2+(s_z-z)^2}\\ &+\sqrt{(p_x-x)^2+(p_y-y)^2+(p_z-z)^2}\\ &-\sqrt{(s_x-p_x)^2+(s_y-p_y)^2+(s_z-p_z)^2}\Big).\end{aligned} \tag{3.9}$$

The square roots in equation (3.9) are divided by N, so that the width of the kernel does not depend on the discretization. The parameter w is the width

of the kernel in arbitrary units and is called *width* in Matlab. The parameter α influences the exponential decay and is called *alpha* in Matlab. A high α gives a high decay, i.e. a more narrow kernel. The coordinates x and z are the coordinates for each center of the voxels. The default values of *width* and *alpha* are chosen because they give a sparsity of the matrix $\mathbf{A}$ of about 5%. To read more about the structure of the matrix $\mathbf{A}$ see section 3.



(a) Solid kernel **(b)** Hollow kernel

Figure 3.5: Illustration of the two kernel types.

Examples:

The problem is set up with a subsurface volume of size $100 \times 100 \times 100$, that is, $N = 100$, and 10 sources and 15 receivers.

```
N = 100;
s = 10;
p = 15;
[A,b,x,sxyz,pxyz] = fresneltomo3D(N,s,p);
```

The phantom, $\mathbf{x}$, and the sources and receivers can be visualized with Sliceomatic [6] in the following code,

```
sliceomatic(flipdim(reshape(x,N,N,N),3))
hold on
plot3(sxyz(:,1),sxyz(:,2),N-sxyz(:,3),'g*','MarkerSize',20,...
'linewidth',2);
hold on
```

```
plot3(pxyz(:,1),pxyz(:,2),N-pxyz(:,3),'mx','MarkerSize',20,...
'linewidth',2);
```

and are displayed in Figure 3.6. The number of source holes is changed by

```
options.sourceHoles = 2;
[A,b,x,sxyz,pxyz] = fresneltomo3D(N,s,p,options)
```

The other fields in *options* can be changed and applied in the same way.

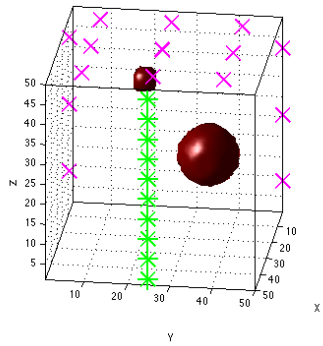


Figure 3.6: Exact phantom with sources (*) and receivers (x), phantom example 1.

See also:

`geophantoms`, `fresneltomo2D`, `tomo3Ddemo` (script)

Limitations:

If *width* is set too high, above 1, and *alpha* is set too low, below 2, then **A** will not be sparse and the function will be slow.

References:

Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*, SIAM, 2010.

Jensen, J. M., B. H. Jacobsen, and J. Christensen-Dalsgaard, *Sensitivity kernels for Time-Distance inversion*, Solar Phys.: SOHO9 topical issue (2000), pp. 231-239

Chapter 4

Image Deblurring Problems

When taking a picture with a camera, it is inevitable that it gets more or less blurred. This means that it is important to have deblurring algorithms to recreate sharp images that are a good representation of the actual scene. In this section, we will set up a test problem for the deblurring problem based on [3].

4.1 Blurring an Image

A digital image is represented by a number of pixels, that is, small squares assigned values representing a color. Here, we will only consider gray scale images, which means that each pixel can be represented by one number between 0 and 1. The blurring of digital images is, that one point in the scene affects not only the pixel representing it, but also neighboring pixels. Deblurring an image is solving an inverse problem (1.1).

4.1.1 Point Spread Functions and Boundary Conditions

A point spread function (PSF) is introduced to model the blurring of the picture. It models how one single pixel will be blurred over the surrounding pixels, and

it can either be represented by a function or an array. The array can be the same size as the image or smaller. If a point spread function array has even dimensions, then the center is placed in the pixel to the right of and below the actual middle of the array. If only one of the dimensions is even, this only applies to that one. If it has uneven dimensions, then the center pixel is at the actual middle.
In this expansion pack four point spread functions have been implemented:

Motion blur: The first point spread function is motion blur. It represents the blurring occurring when the camera is moved while taking a picture, that is, the blurring along a straight line. This can be seen in Figure 4.1a, where the center pixel is blurred to both sides.

Out of focus: The second point spread function represents the camera being out of focus. It is modeled by averaging a pixel value in a disk shape as shown in Figure 4.1b.

Gaussian: The third implemented is a Gaussian point spread function. It blurs a pixel value by the rotationally symmetric Gaussian function, this means that the elements, p_{ij}, in the point spread function array are of the form,

$$p_{ij} = \exp\left(-\frac{1}{2} \begin{bmatrix} i-k \\ j-l \end{bmatrix}^T \begin{bmatrix} s^2 & 0 \\ 0 & s^2 \end{bmatrix}^{-1} \begin{bmatrix} i-k \\ j-l \end{bmatrix} \right),$$

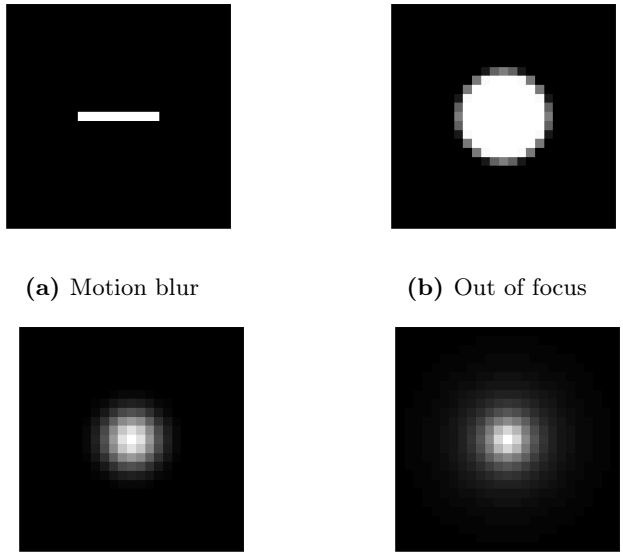
where i and j are the indices in the PSF array, k and l is the center of the PSF array, where a point source is located. The parameter s is the standard deviation. This can be seen in Figure 4.1c.

Moffat: The fourth and final point spread function is the Moffat function. It is mostly used in astronomy and models the blurring of the stars when seen through a telescope. The elements, p_{ij}, of the point spread function array is given as,

$$p_{ij} = \left(1 + \begin{bmatrix} i-k \\ j-l \end{bmatrix}^T \begin{bmatrix} s_1^2 & r^2 \\ r^2 & s_2^2 \end{bmatrix}^{-1} \begin{bmatrix} i-k \\ j-l \end{bmatrix} \right)^{-\beta},$$

where i and j are the indices in the PSF array, k and l is the center of the PSF array where a point source is located. The three parameters s_1, s_2, and r determine the width and orientation of the point spread function. And this can be seen in Figure 4.1d.

Blurring the image means that information about objects near the edge of the image is lost outside of the boundary. It also means that objects just outside the boundary can affect the image. This leads to the need of boundary conditions,



(a) Motion blur **(b)** Out of focus

(c) Gaussian **(d)** Moffat

Figure 4.1: Point Spread Function arrays.

both for blurring the images for test problems, but also for recovering the sharp images. The boundary conditions implemented here, see section 4.2, are zero, periodic, and reflexive boundaries. For blurring the images, we also have the option of using the actual boundaries given by an image.
The condition of zero boundaries means that the image is padded with zeros around the edges. Periodic boundaries mean that the image is repeated beside itself, such that, what is outside the boundary is what is inside the boundary in the opposite side of the image. Finally reflexive boundaries mean that the image is mirrored in the boundary, and the mirrored image is used as the boundary condition.

This leads to setting up the linear system of equations (1.1). The exact image is vectorized and stored in $\mathbf{x}$, and $\mathbf{b}$ is the vectorized blurred image. It is blurred using the PSF array and the boundary condition, which is structured in $\mathbf{A}$.

4.1.2 Applying the PSF: 2-D Convolution

Blurring an image is a discrete 2-D convolution between the exact image and the point spread function array as explained in chapter 4 in [3]. The convolution is most easily explained with an example. First we set up a 3×3 example, where $\mathbf{X}$ is the sharp image matrix, $\mathbf{P}$ is the PSF array, and $\mathbf{B}$ is the blurred image matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \qquad \mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

We use $\mathbf{X}$ and $\mathbf{P}$ to compute the blurred element b_{22}. We rotate the matrix $\mathbf{P}$ 180 degrees due to the 2-D convolution:

$$\overset{\curvearrowright}{\mathbf{P}} = \begin{bmatrix} p_{33} & p_{32} & p_{31} \\ p_{23} & p_{22} & p_{21} \\ p_{13} & p_{12} & p_{11} \end{bmatrix},$$

and place the PSF matrix on top of $\mathbf{X}$, so the center element of $\overset{\curvearrowright}{\mathbf{P}}$ is on top of the element in $\mathbf{X}$ corresponding to the element in $\mathbf{B}$, we wish to compute. Then we multiply and sum

$$\overset{\curvearrowright}{\mathbf{P}}\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ p_{33} & p_{32} & p_{31} \\ x_{21} & x_{22} & x_{23} \\ p_{23} & p_{22} & p_{21} \\ x_{31} & x_{32} & x_{33} \\ p_{13} & p_{12} & p_{11} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

The element b_{22} then becomes,

$$\begin{aligned} b_{22} = {} & p_{33}x_{11} + p_{32}x_{12} + p_{31}x_{13} \\ & + p_{23}x_{21} + p_{22}x_{22} + p_{21}x_{23} \\ & + p_{13}x_{31} + p_{12}x_{32} + p_{11}x_{33}. \end{aligned}$$

To see how the boundary conditions affect the blurred image, we need to compute an element closer to the edge, for example b_{11}. We again place the rotated

PSF array on top of $\mathbf{X}$,

$$\overset{\frown}{\mathbf{P}}\mathbf{X} = \begin{array}{c} p_{33}\quad p_{32}\quad p_{31} \\ p_{23} \\ \\ p_{13} \\ \end{array} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ & & \\ x_{21} & x_{22} & x_{23} \\ & & \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix},$$

and the element b_{11} then becomes,

$$\begin{aligned} b_{11} = p_{33}\,\_\_ + p_{32}\,\_\_ + p_{31}\,\_\_ \\ + p_{23}\,\_\_ + p_{22}\,x_{11} + p_{21}\,x_{12} \\ + p_{13}\,\_\_ + p_{12}\,x_{21} + p_{11}\,x_{22}, \end{aligned} \tag{4.1}$$

where the empty spaces, represent the influence from the image boundaries. If zero boundaries are chosen, the empty spaces become zeros and the elements close to the boundary will have a low value. If periodic is chosen, they take the value of pixels at the opposite side of the image, and if reflexive is chosen they are the mirrored pixel value. To clarify the last two boundary conditions, the periodic and reflexive matrices are written together with $\overset{\frown}{\mathbf{P}}$ as,

$$\overset{\frown}{\mathbf{P}}_{\text{per}}\mathbf{X} = \begin{array}{c} x_{33}\quad x_{31}\quad x_{32}\quad x_{33} \\ p_{33}\quad p_{32}\quad p_{31} \\ x_{13} \\ p_{23} \\ x_{23} \\ p_{13} \\ x_{33} \end{array} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ & & \\ x_{21} & x_{22} & x_{23} \\ & & \\ x_{31} & x_{32} & x_{33} \end{bmatrix},$$

$$\widetilde{\mathbf{P}}_{\text{ref}}\mathbf{X} = \begin{matrix} x_{11} \\ p_{33} \\ \\ x_{11} \\ p_{23} \\ \\ x_{21} \\ p_{13} \\ \\ x_{31} \end{matrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ p_{32} & p_{31} & \\ x_{11} & x_{12} & x_{13} \\ p_{22} & p_{21} & \\ x_{21} & x_{22} & x_{23} \\ p_{12} & p_{11} & \\ x_{31} & x_{32} & x_{33} \end{bmatrix}.$$

The structures of $\mathbf{A}$ therefore depend on both the point spread function and the boundary condition. Each row of $\mathbf{A}$ multiplied by $\mathbf{x}$ then results in an element of $\mathbf{b}$. The structure of $\mathbf{A}$ therefore needs to give the convolution shown above. More about these structures can be found in appendix C.

4.2 imageblur

Purpose:

Test problem: Image deblurring problem.

Synopsis:

```
[A,b,x,m,n] = imageblur()
[A,b,x,m,n] = imageblur(X)
[A,b,x,m,n] = imageblur(X,options)
```

Description:

Input Parameters	
X	A gray scale image. If no input or an empty matrix is given then the default image is used, that is, the small standard image.
options	Struct containing the option fields for the test problem, see below.

Output Parameters	
A	The system matrix of size $(m \cdot n) \times (m \cdot n)$.
b	The blurred image as a column vector of length $m \cdot n$, i.e. `B = reshape(b,m,n)`.
x	The sharp image as a column vector of length $m \cdot n$, i.e. `X = reshape(x,m,n)`.
m	Number of rows in the output images **X** and **B**.
n	Number of columns in the output images **X** and **B**.

Option Fields	
standardImage	Indicates the size of the chosen standard image. The options are: 'small', which is 300×400 pixels, 'medium' of 600×800 pixels, and 'large' of 1200×1600 pixels. The default size is small.
psf	The point spread function. The options are: 'motionBlur', 'outOfFocus', 'gaussian', and 'moffat'. The default is 'outOfFocus'.

blurBoundary	The boundary condition used to blur the image to obtain **b**. The options are: 'zero', 'periodic', 'reflexive', and 'true'. The default setting is 'zero'.
modelBoundary	The boundary condition used to construct **A**. The options are: 'zero', 'periodic', and 'reflexive'. If nothing is given, the model boundary is set to *blurBoundary*, that is unless *blurBoundary* is set to 'true', in which case the default is used. The default is 'zero'.
cropSize	The size of the image blurred with true boundary conditions. It is given in the form $[m\ n]$. If *cropSize* is not given, the size is defined based on the PSF size.
theta	A 'motionBlur' option. The parameter *theta* is the angle in degrees in a counterclockwise direction of the blurring line. The default angle is 0.
len	A 'motionBlur' option. The parameter *len* is the length of the blurring line in pixels. The default length is 9 pixels.
radius	An 'outOfFocus' option. The parameter *radius* is the radius of the disk used in out of focus in pixels. The default size is 5 pixels.
sigma	A 'gaussian' option. The parameter *sigma* is the standard deviation of the Gaussian function in pixels and must be a positive scalar. The default is 2.
beta	A 'moffat' option. The parameter *beta* is a positive parameter that controls the decay of the PSF. The default is set to 1.15. The default value is chosen so the distribution has a suitable size.
r	A 'moffat' option. The parameter r determine the width and orientation of the Moffat function together with s. The default value is 0.
s	A 'moffat' option. The parameter s determine the width and orientation of the Moffat distribution together with r. It can either be given as a vector $s = [s_1\ s_2]$ or a scalar, then $s = s_1 = s_2$. The default value is 2.4. The default value is chosen so the distribution has a suitable size.

Table 4.1

The Matlab function `imageblur` creates a system matrix, **A**, the sharp image as a vector, **x**, and the blurred image as a vector, **b**. It furthermore gives the number of rows and columns in the image as output, they are respectively m and n. The vectorized image **x**, given as output, is either the input image or a cropped version of the input image. The image is cropped, when the blurring

boundary option of true boundaries is used, see more below.
The input image can be provided by the user, otherwise one of the standard images is used. If the function is called without inputs, the small standard image is used. If the input $\mathbf{X}$ is given as an empty matrix, this is also the case, unless otherwise specified in the option *standardImage*. In this option, the user can choose between three sizes of the standard image: A small of size 300×400 pixels, a medium of size 600×800 pixels, and a large of size 1200×1600 pixels. These are only used, if the input $\mathbf{X}$ is an empty matrix.

Point Spread Functions: The model used to blur the image is a point spread function, PSF. It is a model of how one pixel is blurred, and it is applied to all pixels in the image. The implemented PSF's are the options *motionBlur*, *outOfFocus*, *gaussian*, and *moffat* blurring. The PSF *motionBlur* models the camera being moved while the picture is taken, and each pixel is blurred as a straight line. The PSF *outOfFocus* models the camera being out of focus and each pixel is averaged over a disk shape. The PSF *gaussian* models the blurring of each pixel as a Gaussian distribution. The PSF *moffat* models the blurring of each pixel as a Moffat distribution, see the beginning of section 4. The Gaussian and the Moffat PSF arrays are cropped, such that, the elements with lowest values are higher than 5% of the highest values. This is done to limit the size of the PSF arrays.

Boundary Conditions: In this function, there are two types of boundary conditions: The blurring boundary, which is used to blur the image and create $\mathbf{b}$, and a model boundary, which is used to create $\mathbf{A}$, and thereby can be used to recreate the sharp image.
For blurring the image, the *blurBoundary* is used. Here, the user can choose between four boundary conditions: Zero boundaries, periodic boundaries, reflexive boundaries, and true boundaries. The zero boundary option, *zero*, sets everything outside the image to zero. The option *periodic* lets the image be repeated besides itself, such that, the pixels outside the image are the same as the pixels in the opposite side of the image. The option *reflexive* sets the the boundary to be a reflection of the image. The last option for blurring boundary is using the true boundaries. This means, that the input image is blurred, but only the center part of the image is returned to the user as output. The center part is of the size specified in the option *cropSize*. If no size is specified, then it is as large as possible without being affected by any boundary condition applied to the large image. If the given crop size is large, then the center part is affected by the zero blurring boundary condition. If this happens a warning is given.
To create the matrix $\mathbf{A}$ the *modelBoundary* is used. It can be one of three boundary conditions: Zero boundaries, periodic boundaries, or reflexive boundaries. The matrix is created to perform the blurring using the boundary and PSF specified. To see more about the structure of the matrix $\mathbf{A}$, see appendix C.

If only one of the two types of boundary conditions is given by the user, the other is set to the same, unless the blurring boundary is chosen to be 'true'. If this is the case, the model boundary is set to zero. If none of the boundary conditions are specified the default is zero boundaries.

Remaining Options: The rest of the options for the test problem are options regarding the PSF's. These are: *theta*, *len*, *radius*, *sigma*, *beta*, *r*, and *s*. They are specific for one PSF and for which is shown in Table 4.1.

Examples:

The problem is set up with default parameters.

```
[A,b,x,m,n] = imageblur();
```

The default standard image is shown with the following code,

```
imagesc(reshape(x,m,n))
axis image off
colormap gray
```

and is displayed in Figure 4.2. The blurred image, which is blurred with the default out of focus filter, is visualized and shown in Figure 4.3b:

```
imagesc(reshape(b,m,n))
axis image off
colormap gray
```

The point spread function is changed to motion blur and the blurred image is visualized the same way as before in Figure 4.3a.

```
options.psf = 'motionBlur';
[A,b,x,m,n] = imageblur([],options);
```

The rest of point spread functions applied to the standard image is displayed in Figure 4.3c and 4.3d, that is, respectively the Gaussian function and the Moffat function.

See also:

`imageblurdemo` (script)

Figure 4.2: The standard image.

(a) Motion blur

(b) Out of focus

(c) Gaussian

(d) Moffat

Figure 4.3: The standard image blurred with various point spread functions and zero boundary conditions.

Limitations:

If the user chooses a large PSF array, by setting the length of motion blur, the radius of out of focus, the standard deviation of the Gaussian distribution, or

r and s in the Moffat distribution high, then the system matrix $\mathbf{A}$, will not be very sparse and the function will be slow.
If the image is very large then the function will also be slow.

References:

P. C. Hansen, J. G. Nagy and D. P. O'Leary, *Deblurring Images: Matrices, Spectra and Filtering*, SIAM, Philadelphia, 2006.

Chapter 5

Reflections on the Expansion Pack

This expansion pack to Regularization Tools [4] is a product of a bachelor thesis at the Technical University of Denmark in the spring of 2013. The idea was to design good test problems for testing large-scale regularization algorithms in Matlab. We have focused on making the test problems user-friendly, so users with various backgrounds can use the functions.

To make it possible for a user with only little knowledge in the field to use the function, we have minimized the number of required inputs. This means that the functions in this expansion pack at most requires two inputs, but can be given more. It is also an option to provide a range of parameters. This is done by having most of the input parameters as fields in an option struct. Then the user can choose to set the parameters or ignore the option struct completely. The latter will result in use of the default values. This way, it is easy to use the functions without knowing what the different parameters do. The advanced users however still have the options of tweaking the test problem to fit their particular problem.

The two types of geological test problems use a subsurface phantom, that is generated using a special function. This phantom-function was originally a build-in function in the different test problem functions, but we chose to make

it an independent function to create more flexibility in the uses of the expansion pack. This way, the advanced user can create their own specialised phantoms.

Another thing, we have learned during the project is regarding the output parameters. In the beginning of the project, all the test problem functions only had $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}$ as outputs. However, throughout this project it became clear that some of the functions should have more output parameters, some examples are the coordinates for the measurements in the gravity surveying problems and the coordinates for the sources and receivers in the seismic tomography problems. This makes it easier for the user to visualize the problems and understand the physical structure.

For each test problem function there is a demo script related to it. These demo scripts are intended as further introduction on how to use the functions and visualize the outputs and make it easy to begin using our functions. In the demo scripts it is possible to change the setup of a problem without having to write everything from scratch.

In the implementation of the test problems, we have focused on making it possible to run on a laptop computer. This means minimizing the memory space needed for storing the problems, as well as efficient code. We have used sparse matrix structure to store the system matrices for the image deblurring problem and the seismic tomography problem. This is done, because the matrices are intended as being sparse. The default parameters have all been chosen, such that, the system matrices have a sparsity of around 5%. If the parameters are of the recommended size, the sparsity will be at most around 10%. It is however possible to make full matrices, but doing this in a sparse matrix structure, will make the functions run slow.

We hope, that if Regularization Tools is expanded further with more linear test problems, these problems will follow same structure as in this pack, since a lot of thought have gone into structuring the problems in a user-friendly and logical setup. This means, that the output parameters will always be $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}$ together with a few parameters needed for visualization. The input parameters should be as few as possible, but with further parameters in an option struct for the advanced user.

Appendix A

More Geo-phantoms

In this section, the interested reader can find the rest of the formulas referred to in section 2.1, which is the ellipsoid and the Gaussian function i 3-D.

The ellipsoid formula in 3-D with a center in $(0, 0, 0)$ can be found in [10] and is

$$\frac{\tilde{x}^2}{a_1^2} + \frac{\tilde{y}^2}{a_2^2} + \frac{\tilde{z}^2}{a_3^2} \leq 1. \tag{A.1}$$

The ellipsoid has the semi-axes a_1, a_2, and a_3 and $(\tilde{x}, \tilde{y}, \tilde{z})$ are the translated and rotated ellipsoid coordinates. The rotational parameters θ_1, θ_2, and θ_3 are the angles in radians the ellipsoid is rotated around the x, y and z-axes respectively, and is rotated in a left-hand coordinate system using the rotation matrices,

$$R_x(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) \\ 0 & \sin(\theta_1) & \cos(\theta_1) \end{bmatrix}, \tag{A.2}$$

$$R_y(\theta_2) = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix}, \tag{A.3}$$

$$R_z(\theta_3) = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.4}$$

The relation between $(\tilde{x}, \tilde{y}, \tilde{z})$ and the coordinates (x, y, z) is,

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = R_z(\theta_3) R_y(\theta_2) R_x(\theta_1) \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}. \tag{A.5}$$

The (x, y, z), whose translated and rotated coordinates $(\tilde{x}, \tilde{y}, \tilde{z})$ satisfy (A.1), are assigned the material value *rho* to obtain the ellipsoid object. The ball is an ellipsoid with $a_1 = a_2 = a_3$.

The 3-D Gaussian function is implemented with $\sigma = [\sigma_1, \sigma_2, \sigma_3]$ being the variance in the x, y, and z-directions. The correlations are $r = [r_{xy}, r_{xz}, r_{yz}]$; r_{xy} is the xy-correlation, r_{xz} is the xz-correlation, and r_{yz} is the yz-correlation. The mean is $\mu = [\mu_1, \mu_2, \mu_3]$. The Gaussian density function in 3-D is, a generalization of [9],

$$f(x, y, z) = \frac{1}{(2\pi)^{3/2}\sqrt{\det(\Sigma)}} \exp\Big(-\frac{1}{2}(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) \Big), \tag{A.6}$$

where $\mathbf{x} = [x, y, z]$ and Σ is,

$$\Sigma = \begin{bmatrix} \sigma_1^2 & r_{xy}\sigma_1\sigma_2 & r_{xz}\sigma_1\sigma_3 \\ r_{xy}\sigma_1\sigma_2 & \sigma_2^2 & r_{yz}\sigma_2\sigma_3 \\ r_{xz}\sigma_1\sigma_3 & r_{yz}\sigma_2\sigma_3 & \sigma_3^2 \end{bmatrix}.$$

The Gaussian density function is then normalized and assigned the value of *rho* in the center, and the density then decays outward.

Appendix B

Gravity Surveying Model in Three Dimensions

In this appendix, we will derive the system for the three dimensional gravity surveying problem, as we did for the two dimensional problem in section 2. As in 2-D, we have from physics that the gravitational force due to a point mass is,

$$\mathbf{g} = -GM\frac{\hat{\mathbf{r}}}{\|\mathbf{r}\|_2^2}, \tag{B.1}$$

where G is the gravitational constant, M is the strength of the point mass, $\mathbf{r}$ is the vector from the measuring point to the point mass, and $\hat{\mathbf{r}}$ is the unit vector, $\hat{\mathbf{r}} = \mathbf{r}/\|\mathbf{r}\|_2$. The only difference from the 2-D problem is that the vectors in equation (B.1) are now three dimensional. As in section 2, we will neglect the gravitational constant and reverse the vertical axis to get rid of the minus sign. Reversing the axis means that we get a left-hand coordinate system. Again, we are not only interested in one point mass, but many point masses so we need to integrate over the subsurface.

We want to compute the vertical gravity field at a point, $s = (s_1, s_2)$, on the surface. We consider the contribution to the gravity field, dg, from a source point, (x, y, z), below the surface with the mass density $f(x, y, z)$. The distance between the point s on the surface and the source point (x, y, z) is $\|\mathbf{r}\|_2 = \sqrt{z^2 + (s_1 - x)^2 + (s_2 - y)^2}$. Since we are only interested in the vertical

component of the gravity field, we only need the vertical component of the unit vector $\hat{\mathbf{r}}$, that is, $\frac{z}{\|\mathbf{r}\|_2}$. We insert all this into equation (B.1),

$$dg = \frac{\frac{z}{\|\mathbf{r}\|_2}}{\|\mathbf{r}\|_2^2} = \frac{z}{\|\mathbf{r}\|_2^3} f(x,y,z) = \frac{z}{(z^2 + (s_1 - x)^2 + (s_2 - y)^2)^{3/2}} f(x,y,z).$$

We notice that the formula depends on the mass density and not the mass. The mass distribution is found by integrating over the subsurface. The vertical component of the gravitational field at a point s then becomes,

$$g(s) = \int_0^1 \int_0^{d_y} \int_0^{d_z} \frac{z}{(z^2 + (s_1 - x)^2 + (s_2 - y)^2)^{3/2}} f(x,y,z) dz dy dx, \quad \text{(B.2)}$$

where 1 is the length in the x-direction, d_y is the width in the y-direction, and d_z is the depth in the z-direction all of the subsurface volume. The equation (B.2) is a first kind Fredholm integral with the kernel

$$K(s,x,y,z) = \frac{z}{(z^2 + (s_1 - x)^2 + (s_2 - y)^2)^{3/2}}.$$

As in 2-D, we are not interested in the continuous representation, but a discrete one. We therefore need to discretize the subsurface into cubic voxels with N_x in the x-direction, N_y in the y-direction, and N_z in the z-direction. We use the midpoint method from [8] and get,

$$x_{k_1} = \frac{k_1 - \frac{1}{2}}{N_x}, \quad y_{k_2} = \frac{(k_2 - \frac{1}{2})d_y}{N_y}, \quad z_{k_3} = \frac{(k_3 - \frac{1}{2})d_z}{N_z},$$

where $k_1 = 1 \dots N_x, k_2 = 1 \dots N_y, k_3 = 1 \dots N_z$. This leads to the following kernel that depends on the indices,

$$K(s,x,y,z) \approx K_{k_1,k_2,k_3}(s) = \frac{z_{k_3}}{(z_{k_3}^2 + (s_1 - x_{k_1})^2 + (s_2 - y_{k_2})^2)^{3/2}},$$

which furthermore leads to the following discretization of the integral (B.2),

$$\int_0^1 \int_0^{d_y} \int_0^{d_z} K(s) f(x,y,z) dz dy dx, \tag{B.3a}$$

$$\approx \int_0^1 \int_0^{d_y} \frac{d_z}{N_z} \sum_{k_3=1}^{N_z} K_{k_3}(s) f(x,y,z_{k_3}) dy dx, \tag{B.3b}$$

$$\approx \int_0^1 \frac{d_y}{N_y} \frac{d_z}{N_z} \sum_{k_2=1}^{N_y} \sum_{k_3=1}^{N_z} K_{k_2,k_3}(s) f(x,y_{k_2},z_{k_3}) dx, \tag{B.3c}$$

$$\approx \frac{1}{N_x} \frac{d_y}{N_y} \frac{d_z}{N_z} \sum_{k_1=1}^{N_x} \sum_{k_2=1}^{N_y} \sum_{k_3=1}^{N_z} K_{k_1,k_2,k_3}(s) f(x_{k_1},y_{k_2},z_{k_3}), \tag{B.3d}$$

$$= \frac{d_y \cdot d_z}{n} \sum_{k_1=1}^{N_x} \sum_{k_2=1}^{N_y} \sum_{k_3=1}^{N_z} K_{k_1,k_2,k_3}(s) f(x_{k_1},y_{k_2},z_{k_3}), \tag{B.3e}$$

$$= \tilde{g}(s), \tag{B.3f}$$

where $n = N_x \cdot N_y \cdot N_z$. We discretize the measurement interval into a grid of measurement points with m_x points in the x-direction and m_y points in the y-direction. The measurement points are of the form $\tilde{s} = (s_x, s_y)$ and the z-coordinate is zero. We let $b_i = \tilde{g}(\tilde{s})$ for $i = 1 \ldots (m_x \cdot m_y)$.

We can now formulate the problem as the system (1.1), $\mathbf{Ax}$=$\mathbf{b}$, where $\mathbf{x}$ is a vector of the $f(x,y,z)$ values in each voxel of the phantom. The elements of $\mathbf{A}$ are given as,

$$a_{i,j} = \frac{d_y \cdot d_z}{n} \frac{z_{k_3}}{(z_{k_3}^2 + (s_x - x_{k_1})^2 + (s_y - y_{k_2})^2)^{3/2}},$$

where $j = k_1 + (k_2 - 1)N_x + (k_3 - 1)N_x N_y$. The right-hand side $\mathbf{b}$ is a vector containing the elements b_i.

Appendix C

Image Deblurring System Matrix

The structure matrix, $\mathbf{A}$, for image deblurring problems are of the size $(m \cdot n) \times (m \cdot n)$, for an image of size $m \times n$ and depends on the chosen model boundary condition. The structure of $\mathbf{A}$ has to have a form, such that, the matrix multiplication, $\mathbf{Ax} = \mathbf{b}$, is equivalent to a two dimensional convolution between $\mathbf{X}$ and the PSF array, and is best explained by an illustrative example. We have designed the test problem to be able to take in non-quadratic images, we have therefore investigated how the structures of $\mathbf{A}$ look for such problems. A simpler example of a 3×3 image can be found in [3]. Let us look at a 3×5 example:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \end{bmatrix}, \qquad \mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \end{bmatrix}.$$

To set up the problem $\mathbf{Ax} = \mathbf{b}$, the images $\mathbf{X}$ and $\mathbf{B}$ must be vectors. Therefore they are vectorized, and look like

$$\mathbf{x} = \left[\ x_{11}\, x_{21}\, x_{31} \middle|\ x_{12}\, x_{22}\, x_{32} \middle|\ x_{13}\, x_{23}\, x_{33} \middle|\ x_{14}\, x_{24}\, x_{34} \middle|\ x_{15}\, x_{25}\, x_{35}\ \right]^\top, \quad \text{(C.1)}$$

$$\mathbf{b} = \left[\ b_{11}\, b_{21}\, b_{31} \middle|\ b_{12}\, b_{22}\, b_{32} \middle|\ b_{13}\, b_{23}\, b_{33} \middle|\ b_{14}\, b_{24}\, b_{34} \middle|\ b_{15}\, b_{25}\, b_{35}\ \right]^\top. \quad \text{(C.2)}$$

The rotated point spread function matrix becomes,

$$\overset{\curvearrowright}{\mathbf{P}} = \begin{bmatrix} p_{35} & p_{34} & p_{33} & p_{32} & p_{31} \\ p_{25} & p_{24} & p_{23} & p_{22} & p_{21} \\ p_{15} & p_{14} & p_{13} & p_{12} & p_{11} \end{bmatrix}.$$

We know from section 4 that when applying $\overset{\curvearrowright}{\mathbf{P}}$ to $\mathbf{X}$, element b_{11} looks like,

$$\begin{aligned} b_{11} = {} & p_{35}\,\_\_ + p_{34}\,\_\_ + p_{33}\,\_\_ + p_{32}\,\_\_ + p_{31}\,\_\_ \\ & + p_{25}\,\_\_ + p_{24}\,\_\_ + p_{23}\, x_{11} + p_{22}\, x_{12} + p_{21}\, x_{13} \\ & + p_{15}\,\_\_ + p_{14}\,\_\_ + p_{13}\, x_{21} + p_{12}\, x_{22} + p_{11}\, x_{23}. \end{aligned} \quad \text{(C.3)}$$

The empty spaces represent the influence from the boundary condition. This influence defines the structure of $\mathbf{A}$. Each row, i, in $\mathbf{A}$ is the weights of the PSF for a given pixel, b_i, in the blurred image. Each column, j, in $\mathbf{A}$ is the weight of how x_j in $\mathbf{x}$ influence $\mathbf{b}$. When multiplying $\mathbf{A}$ and $\mathbf{x}$, the inner product becomes the blurred image vector $\mathbf{b}$. Because the different boundary conditions define how $\mathbf{A}$ looks like, we will consider the different model boundaries individually. That is, zero boundaries, periodic boundaries, and reflexive boundaries.

C.1 Zero Boundary Condition

Zero boundary condition is when the image $\mathbf{X}$ is padded with zeros around the boundaries, more about this can be found in section 4.1.1. The structure of $\mathbf{A}$ is found by computing each element in $\mathbf{b}$ as done for b_{11} in equation (C.3). To find the pattern of $\mathbf{A}$, we will compute the first couple of rows. The empty spaces in (C.3) are replaced with zeros. We then get:

$$b_{11} = p_{23}\, x_{11} + p_{22}\, x_{12} + p_{21}\, x_{13} + p_{13}\, x_{21} + p_{12}\, x_{22} + p_{11}\, x_{23}.$$

This becomes the first inner product between the first row of $\mathbf{A}$ and $\mathbf{x}$. The next couple of elements of $\mathbf{b}$ take the form:

$$b_{21} = p_{33}\, x_{11} + p_{32}\, x_{12} + p_{31}\, x_{13} + p_{23}\, x_{21} + p_{22}\, x_{22} + p_{21}\, x_{23} + p_{13}\, x_{31} + p_{12}\, x_{32} + p_{11}\, x_{33},$$

$$b_{31} = p_{33}\, x_{21} + p_{32}\, x_{22} + p_{31}\, x_{23} + p_{23}\, x_{31} + p_{22}\, x_{32} + p_{21}\, x_{33}.$$

The computation of $\mathbf{b}$ continues until you can insert the p_{ij} into $\mathbf{A}$, in such a way that $\mathbf{Ax} = \mathbf{b}$. For zero boundaries $\mathbf{A}$ has the form,

$$\left[\begin{array}{ccccc}
\begin{array}{cc} p_{23} & p_{13} \\ p_{33} & p_{23}\, p_{13} \\ & p_{33}\, p_{23} \end{array} &
\begin{array}{cc} p_{22}\, p_{12} \\ p_{32}\, p_{22}\, p_{12} \\ p_{32}\, p_{22} \end{array} &
\begin{array}{cc} p_{21}\, p_{11} \\ p_{31}\, p_{21}\, p_{11} \\ p_{31}\, p_{21} \end{array} & & \\
\hline
\begin{array}{cc} p_{24} & p_{14} \\ p_{34} & p_{24}\, p_{14} \\ & p_{34}\, p_{24} \end{array} &
\begin{array}{cc} p_{23}\, p_{13} \\ p_{33}\, p_{23}\, p_{13} \\ p_{33}\, p_{23} \end{array} &
\begin{array}{cc} p_{22}\, p_{12} \\ p_{32}\, p_{22}\, p_{12} \\ p_{32}\, p_{22} \end{array} &
\begin{array}{cc} p_{21}\, p_{11} \\ p_{31}\, p_{21}\, p_{11} \\ p_{31}\, p_{21} \end{array} & \\
\hline
\begin{array}{cc} p_{25} & p_{15} \\ p_{35} & p_{25}\, p_{15} \\ & p_{35}\, p_{25} \end{array} &
\begin{array}{cc} p_{24}\, p_{14} \\ p_{34}\, p_{24}\, p_{14} \\ p_{34}\, p_{24} \end{array} &
\begin{array}{cc} p_{23}\, p_{13} \\ p_{33}\, p_{23}\, p_{13} \\ p_{33}\, p_{23} \end{array} &
\begin{array}{cc} p_{22}\, p_{12} \\ p_{32}\, p_{22}\, p_{12} \\ p_{32}\, p_{22} \end{array} &
\begin{array}{cc} p_{21}\, p_{11} \\ p_{31}\, p_{21}\, p_{11} \\ p_{31}\, p_{21} \end{array} \\
\hline
& \begin{array}{cc} p_{25}\, p_{15} \\ p_{35}\, p_{25}\, p_{15} \\ p_{35}\, p_{25} \end{array} &
\begin{array}{cc} p_{24}\, p_{14} \\ p_{34}\, p_{24}\, p_{14} \\ p_{34}\, p_{24} \end{array} &
\begin{array}{cc} p_{23}\, p_{13} \\ p_{33}\, p_{23}\, p_{13} \\ p_{33}\, p_{23} \end{array} &
\begin{array}{cc} p_{22}\, p_{12} \\ p_{32}\, p_{22}\, p_{12} \\ p_{32}\, p_{22} \end{array} \\
\hline
& & \begin{array}{cc} p_{25}\, p_{15} \\ p_{35}\, p_{25}\, p_{15} \\ p_{35}\, p_{25} \end{array} &
\begin{array}{cc} p_{24}\, p_{14} \\ p_{34}\, p_{24}\, p_{14} \\ p_{34}\, p_{24} \end{array} &
\begin{array}{cc} p_{23}\, p_{13} \\ p_{33}\, p_{23}\, p_{13} \\ p_{33}\, p_{23} \end{array}
\end{array}\right] \tag{C.4}$$

The matrix in (C.4) has a block Toeplitz Toeplitz block structure (BTTB). That is, it has five blocks of 3×3 each with a Toeplitz structure, and they are again placed in a Toeplitz structure. More about Toeplitz matrices can be found in [1].

C.2 Periodic Boundary Condition

Periodic boundary condition is when the pixel values are repeated periodically both horizontally and vertically with a period length of the number of pixels respectively in the rows and columns, more about this can be found in section 4.1.1. When applying periodic boundary conditions, the elements of $\mathbf{b}$ are

computed as in equation (C.3). Let us look at the first couple elements in $\mathbf{b}$,

$$\begin{aligned}
b_{11} &= p_{35}\,x_{34} + p_{34}\,x_{35} + p_{33}\,x_{31} + p_{32}\,x_{32} + p_{31}\,x_{33} \\
&\quad + p_{25}\,x_{14} + p_{24}\,x_{15} + p_{23}\,x_{11} + p_{22}\,x_{12} + p_{21}\,x_{13} \\
&\quad + p_{15}\,x_{24} + p_{14}\,x_{25} + p_{13}\,x_{21} + p_{12}\,x_{22} + p_{11}\,x_{23}, \\
b_{21} &= p_{35}\,x_{14} + p_{34}\,x_{15} + p_{33}\,x_{11} + p_{32}\,x_{12} + p_{31}\,x_{13} \\
&\quad + p_{25}\,x_{24} + p_{24}\,x_{25} + p_{23}\,x_{21} + p_{22}\,x_{22} + p_{21}\,x_{23} \\
&\quad + p_{15}\,x_{34} + p_{14}\,x_{35} + p_{13}\,x_{31} + p_{12}\,x_{32} + p_{11}\,x_{33}, \\
b_{31} &= p_{35}\,x_{24} + p_{34}\,x_{25} + p_{33}\,x_{21} + p_{32}\,x_{22} + p_{31}\,x_{23} \\
&\quad + p_{25}\,x_{34} + p_{24}\,x_{35} + p_{23}\,x_{31} + p_{22}\,x_{32} + p_{21}\,x_{33} \\
&\quad + p_{15}\,x_{14} + p_{14}\,x_{15} + p_{13}\,x_{11} + p_{12}\,x_{12} + p_{11}\,x_{13}.
\end{aligned}$$

This continues, and $\mathbf{A}$ is constructed with the elements of $\mathbf{P}$ arranged in such a way that $\mathbf{Ax} = \mathbf{b}$. The matrix $\mathbf{A}$ for periodic boundaries becomes,

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc}
p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{25} & p_{15} & p_{35} & p_{24} & p_{14} & p_{34} \\
p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{35} & p_{25} & p_{15} & p_{34} & p_{24} & p_{14} \\
p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{15} & p_{35} & p_{25} & p_{14} & p_{34} & p_{24} \\
\hline
p_{24} & p_{14} & p_{34} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{25} & p_{15} & p_{35} \\
p_{34} & p_{24} & p_{14} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{35} & p_{25} & p_{15} \\
p_{14} & p_{34} & p_{24} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{15} & p_{35} & p_{25} \\
\hline
p_{25} & p_{15} & p_{35} & p_{24} & p_{14} & p_{34} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} \\
p_{35} & p_{25} & p_{15} & p_{34} & p_{24} & p_{14} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\
p_{15} & p_{35} & p_{25} & p_{14} & p_{34} & p_{24} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} \\
\hline
p_{21} & p_{11} & p_{31} & p_{25} & p_{15} & p_{35} & p_{24} & p_{14} & p_{34} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} \\
p_{31} & p_{21} & p_{11} & p_{35} & p_{25} & p_{15} & p_{34} & p_{24} & p_{14} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\
p_{11} & p_{31} & p_{21} & p_{15} & p_{35} & p_{25} & p_{14} & p_{34} & p_{24} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} \\
\hline
p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{25} & p_{15} & p_{35} & p_{24} & p_{14} & p_{34} & p_{23} & p_{13} & p_{33} \\
p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{35} & p_{25} & p_{15} & p_{34} & p_{24} & p_{14} & p_{33} & p_{23} & p_{13} \\
p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{15} & p_{35} & p_{25} & p_{14} & p_{34} & p_{24} & p_{13} & p_{33} & p_{23}
\end{array}\right]. \quad \text{(C.5)}$$

The matrix in (C.5) has the structure of five 3×3 blocks each with circulant matrix structure, where the the diagonal and the two bidiagonals are the same as in (C.4), but with added weight in the corners. The meta-structure of these blocks is also circulant. More about circulant matrices can be found in [1].

C.3 Reflexive Boundary Condition

Reflexive boundary condition is when the pixel values are mirrored in the edges, see section 4.1.1. This creates a very complex structure matrix that consist of

four different types of matrices. As in the sections C.1 and C.2, we start by computing the first couple of elements of $\mathbf{b}$,

$$\begin{aligned}
b_{11} = {} & p_{35}\, x_{12} + p_{34}\, x_{11} + p_{33}\, x_{11} + p_{32}\, x_{12} + p_{31}\, x_{13} \\
& + p_{25}\, x_{12} + p_{24}\, x_{11} + p_{23}\, x_{11} + p_{22}\, x_{12} + p_{21}\, x_{13} \\
& + p_{15}\, x_{22} + p_{14}\, x_{21} + p_{13}\, x_{21} + p_{12}\, x_{22} + p_{11}\, x_{23},
\end{aligned}$$

$$\begin{aligned}
b_{21} = {} & p_{35}\, x_{12} + p_{34}\, x_{11} + p_{33}\, x_{11} + p_{32}\, x_{12} + p_{31}\, x_{13} \\
& + p_{25}\, x_{22} + p_{24}\, x_{21} + p_{23}\, x_{21} + p_{22}\, x_{22} + p_{21}\, x_{23} \\
& + p_{15}\, x_{32} + p_{14}\, x_{31} + p_{13}\, x_{31} + p_{12}\, x_{32} + p_{11}\, x_{33},
\end{aligned}$$

$$\begin{aligned}
b_{31} = {} & p_{35}\, x_{22} + p_{34}\, x_{21} + p_{33}\, x_{21} + p_{32}\, x_{22} + p_{31}\, x_{23} \\
& + p_{25}\, x_{32} + p_{24}\, x_{31} + p_{23}\, x_{31} + p_{22}\, x_{32} + p_{21}\, x_{33} \\
& + p_{15}\, x_{32} + p_{14}\, x_{31} + p_{13}\, x_{31} + p_{12}\, x_{32} + p_{11}\, x_{33}.
\end{aligned}$$

Here, we see that each element in $\mathbf{A}$ becomes a sum of several PSF weights. If we rearrange them according to x_j it is even clearer and we get:

$$\begin{aligned}
b_{11} = {} & x_{11}\,(p_{34} + p_{33} + p_{24} + p_{23}) + x_{21}\,(p_{14} + p_{13}) \\
& + x_{12}\,(p_{35} + p_{32} + p_{25} + p_{22}) + x_{22}\,(p_{15} + p_{12}) \\
& + x_{13}\,p_{21} + x_{23}\,p_{11},
\end{aligned}$$

$$\begin{aligned}
b_{21} = {} & x_{11}\,(p_{34} + p_{33}) + x_{21}\,(p_{24} + p_{23}) + x_{31}\,(p_{14} + p_{13}) \\
& + x_{12}\,(p_{35} + p_{32}) + x_{22}\,(p_{25} + p_{22}) + x_{32}\,(p_{15} + p_{12}) \\
& + x_{13}\,p_{31} + x_{23}\,p_{21} + x_{33}\,p_{11},
\end{aligned}$$

$$\begin{aligned}
b_{31} = {} & x_{21}\,(p_{34} + p_{33}) + x_{31}\,(p_{24} + p_{23} + p_{14} + p_{13}) \\
& + x_{22}\,(p_{35} + p_{32}) + x_{32}\,(p_{25} + p_{22} + p_{15} + p_{12}) \\
& + x_{23}\,p_{31} + x_{33}\,(p_{21} + p_{11}).
\end{aligned}$$

This continuous until all the elements have been computed. Then we have separated the weights according to their structure. We find that as for the quadratic matrices in [3], $\mathbf{A}$ is a sum of a block Toeplitz Toeplitz block (BTTB) matrix, a block Toeplitz Hankel block (BTHB) matrix, a block Hankel Toeplitz block (BHTB) matrix and finally a block Hankel Hankel block (BHHB) matrix. More about these matrix structures can be found in [1]. The four matrices written out can be seen respectively in (C.6), (C.7), (C.8), and (C.9). The

matrix $\mathbf{A}$ for reflexive boundaries is the sum of the following four matrices,

$$
\left[\begin{array}{ccccc}
p_{23}\ p_{13} & p_{22}\ p_{12} & p_{21}\ p_{11} & & \\
p_{33}\ p_{23}\ p_{13} & p_{32}\ p_{22}\ p_{12} & p_{31}\ p_{21}\ p_{11} & & \\
p_{33}\ p_{23} & p_{32}\ p_{22} & p_{31}\ p_{21} & & \\
\hline
p_{24}\ p_{14} & p_{23}\ p_{13} & p_{22}\ p_{12} & p_{21}\ p_{11} & \\
p_{34}\ p_{24}\ p_{14} & p_{33}\ p_{23}\ p_{13} & p_{32}\ p_{22}\ p_{12} & p_{31}\ p_{21}\ p_{11} & \\
p_{34}\ p_{24} & p_{33}\ p_{23} & p_{32}\ p_{22} & p_{31}\ p_{21} & \\
\hline
p_{25}\ p_{15} & p_{24}\ p_{14} & p_{23}\ p_{13} & p_{22}\ p_{12} & p_{21}\ p_{11} \\
p_{35}\ p_{25}\ p_{15} & p_{34}\ p_{24}\ p_{14} & p_{33}\ p_{23}\ p_{13} & p_{32}\ p_{22}\ p_{12} & p_{31}\ p_{21}\ \ p_{11} \\
p_{35}\ p_{25} & p_{34}\ p_{24} & p_{33}\ p_{23} & p_{32}\ p_{22} & p_{31}\ \ p_{21} \\
\hline
 & p_{25}\ p_{15} & p_{24}\ p_{14} & p_{23}\ p_{13} & p_{22}\ p_{12} \\
 & p_{35}\ p_{25}\ p_{15} & p_{34}\ p_{24}\ p_{14} & p_{33}\ p_{23}\ p_{13} & p_{32}\ p_{22}\ \ p_{12} \\
 & p_{35}\ p_{25} & p_{34}\ p_{24} & p_{33}\ p_{23} & p_{32}\ \ p_{22} \\
\hline
 & & p_{25}\ p_{15} & p_{24}\ p_{14} & p_{23}\ p_{13} \\
 & & p_{35}\ p_{25}\ p_{15} & p_{34}\ p_{24}\ p_{14} & p_{33}\ p_{23}\ \ p_{13} \\
 & & p_{35}\ p_{25} & p_{34}\ p_{24} & p_{33}\ \ p_{23}
\end{array}\right] + \qquad \text{(C.6)}
$$

$$
\left[\begin{array}{ccccc}
p_{24}\ p_{14} & p_{25}\ p_{15} & & & \\
p_{34}\ p_{24}\ p_{14} & p_{35}\ p_{25}\ p_{15} & & & \\
p_{34}\ p_{24} & p_{35}\ p_{25} & & & \\
\hline
p_{25}\ p_{15} & & & & \\
p_{35}\ p_{25}\ p_{15} & & & & \\
p_{35}\ p_{25} & & & & \\
\hline
 & & & & \\
 & & & & \\
\hline
 & & & & p_{21}\ p_{11} \\
 & & & & p_{31}\ p_{21}\ p_{11} \\
 & & & & p_{31}\ p_{21} \\
\hline
 & & & p_{21}\ p_{11} & p_{22}\ p_{12} \\
 & & & p_{31}\ p_{21}\ p_{11} & p_{32}\ p_{22}\ p_{12} \\
 & & & p_{31}\ p_{21} & p_{32}\ p_{22}
\end{array}\right] + \qquad \text{(C.7)}
$$

$$\begin{bmatrix}
\begin{matrix} p_{33} & & \\ & & p_{13} \end{matrix} & \begin{matrix} p_{32} & & \\ & & p_{12} \end{matrix} & \begin{matrix} p_{31} & & \\ & & p_{11} \end{matrix} & & \\
\begin{matrix} p_{34} & & \\ & & p_{14} \end{matrix} & \begin{matrix} p_{33} & & \\ & & p_{13} \end{matrix} & \begin{matrix} p_{32} & & \\ & & p_{12} \end{matrix} & \begin{matrix} p_{31} & & \\ & & p_{11} \end{matrix} & \\
\begin{matrix} p_{35} & & \\ & & p_{15} \end{matrix} & \begin{matrix} p_{34} & & \\ & & p_{14} \end{matrix} & \begin{matrix} p_{33} & & \\ & & p_{13} \end{matrix} & \begin{matrix} p_{32} & & \\ & & p_{12} \end{matrix} & \begin{matrix} p_{31} & & \\ & & p_{11} \end{matrix} \\
 & \begin{matrix} p_{35} & & \\ & & p_{15} \end{matrix} & \begin{matrix} p_{34} & & \\ & & p_{14} \end{matrix} & \begin{matrix} p_{33} & & \\ & & p_{13} \end{matrix} & \begin{matrix} p_{32} & & \\ & & p_{12} \end{matrix} \\
 & & \begin{matrix} p_{35} & & \\ & & p_{15} \end{matrix} & \begin{matrix} p_{34} & & \\ & & p_{14} \end{matrix} & \begin{matrix} p_{33} & & \\ & & p_{13} \end{matrix}
\end{bmatrix} + \quad \text{(C.8)}$$

$$\begin{bmatrix}
\begin{matrix} p_{34} & & \\ & & p_{14} \end{matrix} & \begin{matrix} p_{35} & & \\ & & p_{15} \end{matrix} & & & \\
\begin{matrix} p_{35} & & \\ & & p_{15} \end{matrix} & & & & \\
 & & & & \\
 & & & & \begin{matrix} p_{31} & & \\ & & p_{11} \end{matrix} \\
 & & & \begin{matrix} p_{31} & & \\ & & p_{11} \end{matrix} & \begin{matrix} p_{32} & & \\ & & p_{12} \end{matrix}
\end{bmatrix} . \quad \text{(C.9)}$$

Here, it can be noticed that the first matrix (C.6) is the same as the structure matrix (C.4) for zero boundary conditions. The rest of the matrices are due to the fact that the pixel values are mirrored.

Bibliography

[1] Althaus, G.W. and E. Spedicato, *Algorithms for Large Scale Linear Algebraic Systems: Application in Science and Engineering*, NATO ASI, 1996.

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

[3] Hansen, P. C., J. G. Nagy and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, 2006.

[4] Hansen, P. C., *Regularization Tools Version 4.0 for Matlab 7.3*, Numer. Algo., 46 (2007), pp. 189–194.
The current version of the Matlab package can be found at: http://www.mathworks.com/matlabcentral/fileexchange/52

[5] Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*, SIAM, 2010.

[6] Ludlam, E., *Sliceomatic*, http://www.mathworks.com, June 13 2013.
The current version of the Matlab package can be found at: http://www.mathworks.com/matlabcentral/fileexchange/764-sliceomatic

[7] Jensen, J. M., B. H. Jacobsen and J. Christensen-Dalsgaard, *Sensitivity kernels for Time-Distance inversion*, Solar Phys.: SOHO9 topical issue (2000), pp. 231–239.

[8] Mathews, J.H., *Numerical Methods: For Computer Science, Engineering, and Mathematics*, Prentice-Hall, 1987, p. 350.

[9] Pitman, J. *Probability*, Springer, 2006

[10] Spiegel, M.R., S. Lipschutz and J. Liu, *Mathematical Handbook of Formulas and Tables*, McGraw-Hill, 2009.

[11] Vasco, D., J. Jr. Peterson and E. Majer, *Beyond ray tomography: Wavepaths and Fresnel volumes*, GEOPHYSICS, 60(6) (1995), pp. 1790–1804.