Material inversion with SW4mopt

Björn Sjögreen[1] N. Anders Petersson[1]

September 27, 2013

[1]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, PO Box 808, Livermore CA 94551.

Chapter 1

Introduction

SW4mopt is a solver for material inversion and source inversion, built on the forward solver *SW4*. Running *SW4mopt* is very similar to running *SW4*. *SW4mopt* uses the same input commands as *SW4*, and provides an extended set of commands, related to the inverse problem.

Given time series data at a number of stations, *SW4mopt* solves for the material density, ρ, and Lamé parameters μ and λ, in a material that has been parameterized in some way. Currently, the only available parameterization is a representation of the material on a coarse grid. The material at the grid points is defined by interpolation from this coarse grid. *SW4mopt* is written such that it will be easy to extend to other types of parameterizations, e.g., representation as B-splines or a mesh free unstructured representation.

For minimization, *SW4mopt* provides the choice of the limited memory BFGS method (L-BFGS) or a non-linear conjugate gradient method. These are run together with a line search algorithm to determine the step length.

As of writing, September 2013, *SW4mopt* has been successfully run on simple problems with synthetic data, some of them reported below. It is a complete solver for the inverse problem. However, it is far from being ready to be released to the users. *SW4mopt* is not as robust as *SW4*, and we need to test it on many other problems, to make sure that it works as expected, and to gain experience of material inversion. Furthermore, since *SW4mopt* is still in development, it produces a lot of output messages that might be confusing to the average user.

The following should be done to improve *SW4mopt*.

- Implement the material parameterization for grids with topography. Topography is the only critical part that is missing. The material gradient can be computed with topography, it is just the coarse grid parameterization that has not been done.

- Implement a material parameterization which is distributed on the processors. Currently one copy of the entire coarse material grid is stored in each processor. This will lead to memory limitations as problems size increases.

- Possibility to refine the material grid, so that the inversion to a highly resolved material can be made hierarchally. The same holds for the scaling factors. With a higher material resolution, we can not expect to compute these by forming the Hessian. Can scaling factors be interpolated from a coarse grid inversion ?

- Automatic readjustment of the time step when the material speed increases past the largest stable CFL number. This must now be handled by manual restart, it would be better to have the code doing it automatically.

A few more items that should be addressed, but which require new research and development, and new software.

- Material inversion in an anisotropic material.

- Homogenization techniques to recover isotropic material properties from the result of anisotropic material inversion.

- Material inversion with attenuation.

Chapter 2

The material description

2.1 The mparcart command

The `mparcart` defines a material that is discretized on a coarse Cartesian grid, below refered to as the "material grid". The parameters are the offset values of ρ, μ, and λ relative a reference material, at the points of this grid. The material on the computational grid is defined by trilinear interpolation from the values on the coarser material grid. For example, the density, ρ, at a grid point (i, j, k) in the computational grid is

$$\rho_{i,j,k} = I(\{d^{(\rho)}\}, x_i, y_j, z_k) + \rho^{(0)}_{i,j,k}$$

where $\rho^{(0)}$ is the reference material, and $d^{(\rho)}$ is the difference $\rho - \rho^{(0)}$ on the material grid. The interpolation operator

$$I(\{u\}, x, y, z)$$

evaluates a function u defined at the points of the material grid, at the point (x, y, z).

The number of points in the material grid, and initial values for $d^{(\rho)}, d^{(\mu)}, d^{(\lambda)}$ are specified by the `mparcart` command. For example

```
mparcart nx=5 ny=5 nz=3 init=0
```

defines a material grid with $5\times5\times3$ points, and with $d^{(\rho)}$, $d^{(\mu)}$, $d^{(\lambda)}$ initialized to zero at all points. The reference material is specified by one of the material commands of *SW4*, e.g., `block` or `pfile`, see the *SW4* User's Guide for a complete description of these material commands.

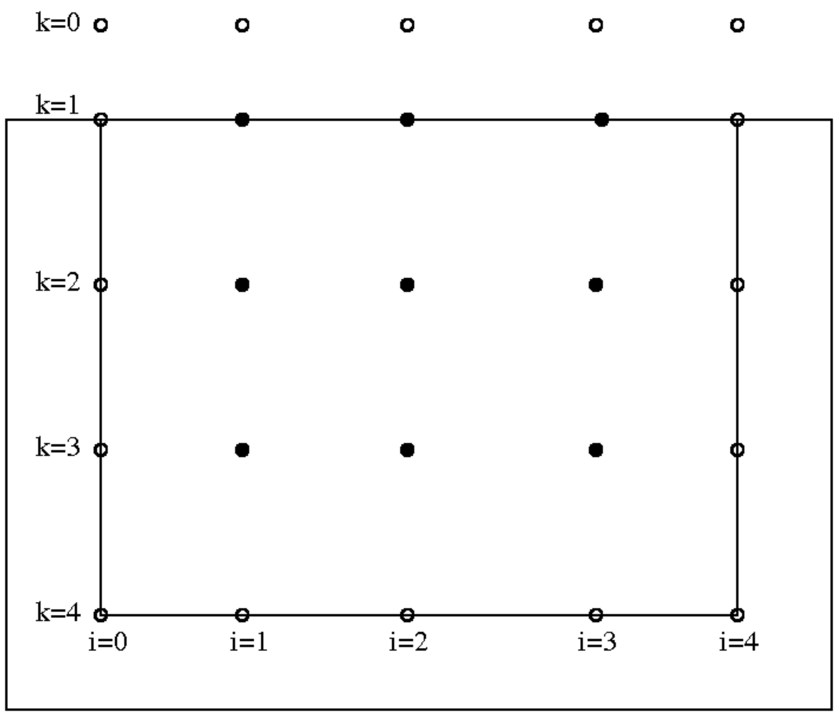


Figure 2.1: Two dimensional slice of a material grid with $n_x = n_y = n_z = 3$. The free surface boundary is at $k = 1$, there are super-grid sponge layers on the other sides. Filled circles indicate unknown parameters, open circles are fixed boundary values.

Using a coarse material grid reduces the number of unknowns, compared with using the material at each grid point as parameters. Furthermore, the resolution in the material is limited in terms of highest frequencies of the computed wave field, making it unreasonable to expect to resolve the material down to the resolution of the computational grid. We recommend that the grid spacing of the material grid is around 10 times the grid spacing of the computational grid. Also to note, the grid spacing of the material grid do not need to have any relation to the spacing of the computational grid. These two grids are independent.

The material grid discretizes only the interior part the domain, without the super-grid sponge layers. The configuration is outlined in Fig. 2.1. The material grid is given by

$$x_i = x_{min} + ih_x \qquad i = 0, \dots, n_x + 1 \tag{2.1}$$

$$y_j = y_{min} + jh_y \qquad j = 0, \dots, n_y + 1 \tag{2.2}$$

$$z_k = z_{min} + kh_z \qquad k = 0, \dots, n_z + 1 \tag{2.3}$$

where the grid spacings h_x, h_y, and h_z are determined such that $x_0, y_0, x_{n_x+1}, y_{n_y+1}$, and z_{n_z+1} are located at the interface between the interior domain and the super-grid sponge layer. In the depth direction, z_1 is on the free surface and, hence $z_{min} = -h_z$. The point z_0 is above the topography, but it will never be used in the interpolation.

At the first and last points in each direction, ($i = 0$, $j = 0$, $k = 0$, $n_x + 1$, $n_y + 1$, and $n_z + 1$), the offsets $d^{(\rho)}$, $d^{(\mu)}$, and $d^{(\lambda)}$ are fixed at zero, hence these values are not part of the parameter vector. The total number of unknowns for the material inversion is $3 \times n_x \times n_y \times n_z$.

Chapter 3

Computed examples

3.1 Material grid with a single point

This test problem is defined on a domain of size $35000 \times 35000 \times 17000$. The material is constant with $\rho = 2650$, $v_s = 2437.56$, and $v_p = 4630.76$, giving $\mu = 1.57455 \times 10^{10}$ and $\lambda = 2.5335 \times 10^{10}$. The elastic wave equation is discretized on a grid with spacing $h = 200$. A traction free boundary condition is imposed at the boundary $z = 0$. All other boundaries have super-grid sponge layers of width 2500. Waves are set off by a moment source located at $x_s = y_s = 17500$, $z_s = 2000$. The only non-zero element of the moment tensor is $M_{xy} = 1.0 \times 10^{18}$. The source time function is a Gaussian with $t_0 = 1.5$ and $\omega = 4$.

We consider a material grid of size $1 \times 1 \times 1$, i.e., there is only one point in the material parameterization. The total number of material parameters is 3, i.e., ρ, μ, and λ at the single point. The outline of the material grid in Fig. 2.1 shows that with a single point, it will be located at $x = y = 17500$ and $z = 0$.

Synthetic seismograms are computed at stations on a centered 5×5 grid with spacing 3000 on the surface, ($z = 0$). These seismograms, computed with the constant material, are used as the measured data in the numerical experiments. In these experiments, we initialize the values at the material grid point by a perturbation of the constant material. Denoting the constant, reference, density by $\rho^{(ref)}$, and the perturbation by $d^{(\rho)}$, we have

$$\rho = \rho^{(ref)} + d^{(\rho)}$$

at the material grid point. The notation for μ and λ are similar. The following five initial perturbations are considered

Name	$d^{(\rho)}$	$d^{(\mu)}$	$d^{(\lambda)}$
pp10	$0.1\rho^{(ref)}$	$0.1\mu^{(ref)}$	$0.1\lambda^{(ref)}$
mm10	$-0.1\rho^{(ref)}$	$-0.1\mu^{(ref)}$	$-0.1\lambda^{(ref)}$
mp10	$-0.1\rho^{(ref)}$	$0.1\mu^{(ref)}$	$0.1\lambda^{(ref)}$
mp30	$-0.3\rho^{(ref)}$	$0.3\mu^{(ref)}$	$0.3\lambda^{(ref)}$
pm30	$0.3\rho^{(ref)}$	$-0.3\mu^{(ref)}$	$-0.3\lambda^{(ref)}$

The minimizing algorithm should converge to the constant material, i.e., $d^{(\rho)}$ should approach zero. The Hessian at the constant material is

$$H = \begin{pmatrix} 2.9371e-03 & -4.9563e-10 & -1.8236e-12 \\ -4.9563e-10 & 8.7013e-17 & 2.4771e-19 \\ -1.8236e-12 & 2.4771e-19 & 3.6478e-20 \end{pmatrix}$$

The condition number of H is 8.571×10^{16}. The scale factors obtained from the diagonal elements of H, and rescaled to have $s_\rho = \rho^{(ref)}$ are,

$$s_\rho = 2650 \quad s_\mu = 1.54 \times 10^{10} \quad s_\lambda = 7.52 \times 10^{11}$$

with these scale factors, the condition number is 107. Note that s_ρ and s_μ are typical sizes of $\rho^{(ref)}$ and $\mu^{(ref)}$ respectively, while s_λ is around 30 times larger than $\lambda^{(ref)}$. As an illustration, Fig. 3.1 shows the density on the plane $x = 17500$ after 1, 2, 3, and 4 iteration with the L-BFGS method. The initial perturbation at the single material grid point gives a hat shaped initial material. The perturbation disappears as the minimizing iterations converges to the constant material used to compute the synthetics. Figure 3.2 shows the convergence histories for the cases in Table 3.1 with 10% perturbation. The convergence from the 30% perturbation are shown in Fig. 3.3.

With the 10% perturbation, it takes 7-10 iterations to converge the misfit down to its final value. The 30% perturbations require 17-20 iterations.

Next, we investigate the convergence of the material properties at the material grid point. Figures 3.4 and 3.5 show the evolution of the relative error, $d^{(\rho)}/\rho^{(ref)}$ for the density and similarly for μ and λ. The perturbation converges to zero, and as shown by the convergence curves, the material has converged in the picture norm after around 10 iterations for the 10% perturbations. With the 30% initial perturbation, around 20 iterations are needed. Figures 3.4 and 3.5 also show that the amplitudes of the perturbations in

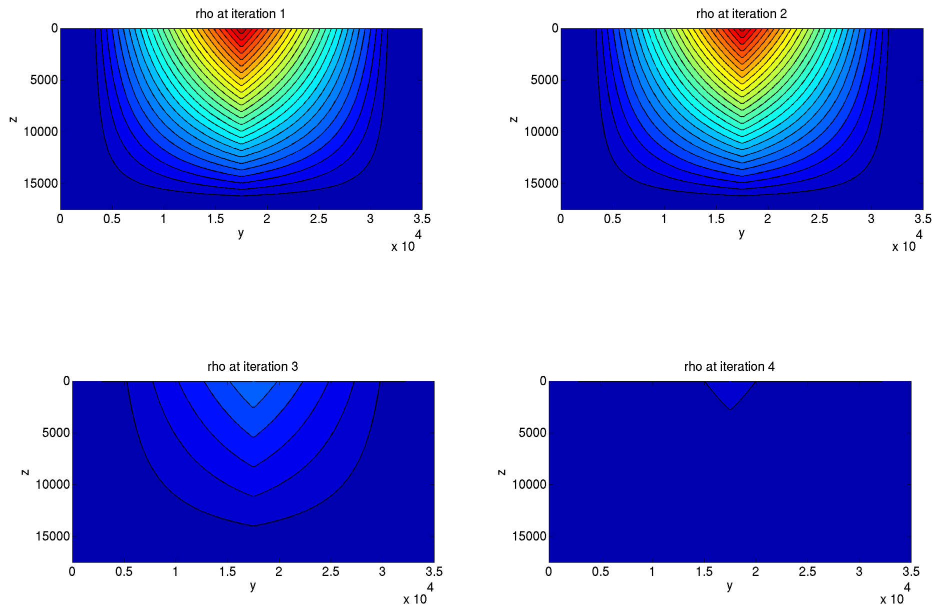


Figure 3.1: Density at $x = 17500$ after iterations 1, 2, 3, and 4. 30 contour levels between 2650 and 2925.

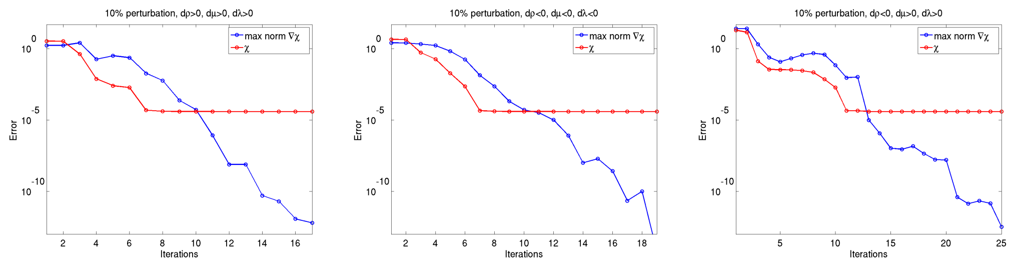
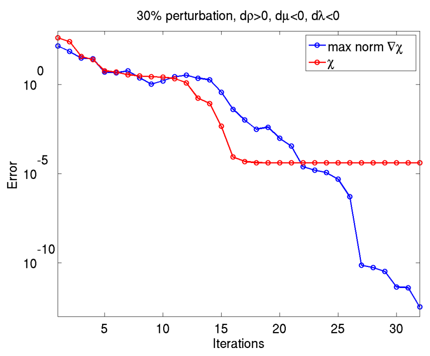


Figure 3.2: Convergence rate of problems pp10 (left), mm10 (middle), and mp10 (right). Blue is the maximum norm of the gradient of the misfit, red is the misfit.



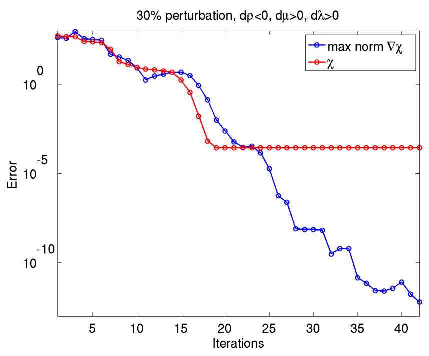
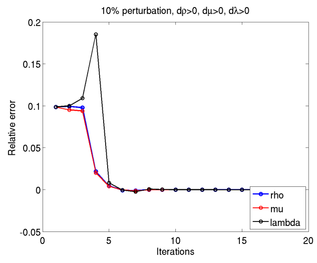
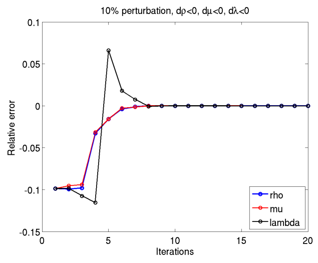


Figure 3.3: Convergence rate of problems pm30 (left), mp30 (right). Blue is the maximum norm of the gradient of the misfit, red is the misfit.





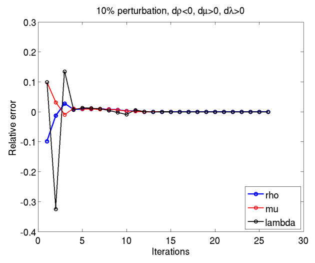


Figure 3.4: Convergence of ρ (blue), μ (red), and λ (black) for problems pp10 (left), mm10 (middle), and mp10 (right).

ρ and μ never exceeds the size of the initial perturbation. λ, on the other hands, have very large variation, especially for the 30% perturbation. This is caused by the scaling factor s_λ being much larger than the size of $\lambda^{(ref)}$. The line search algorithm bases its largest allowed step length on the scale factors, which implies that a much longer relative step is allowed for λ than for ρ and μ.

The input file for this example contains the material specification

```
block vp=4630.76 vs=2437.56 r=2650
mparcart nx=1 ny=1 nz=1 init=onep-pr10.bin
```

meaning that the reference material is defined by the block command, and the initial 10 percent perturbation is read from the file `onep-pr10.bin`. This

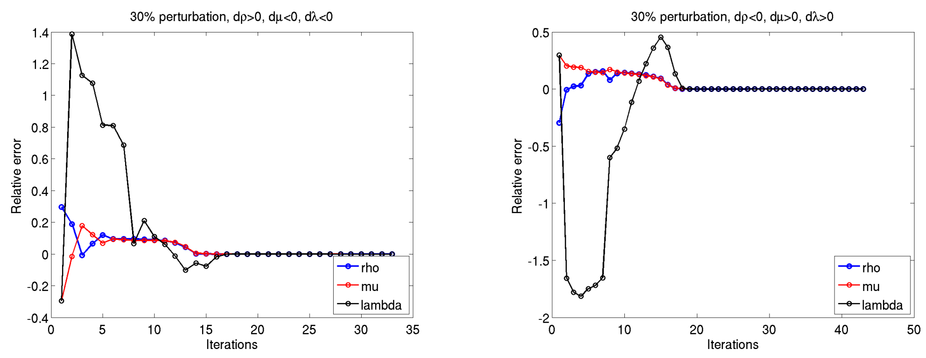


Figure 3.5: Convergence of ρ (blue), μ (red), and λ (black) for problems pm30 (left) and mp30 (right).

file was prepared externally to *SW4mopt*, by a Matlab script. The minimizing algorithm was specified by the lines

```
mrun task=minvert mcheck=on tsoutput=on
lbfgs nvectors=3 maxit=100 tolerance=1e-12 linesearch=on
```

in the inputfile.

3.2 Material with sinusodial variation, $3\times3\times2$ material grid

This test problem has the same dimensions and source as the problem in the previous subsection. The synthetics are computed on a constant material with an added sinusodial variation. The constant material has the properties $\rho = 2650$, $v_s = 2679.5$, and $v_p = 5000$, leading to $\mu = 1.90 \times 10^{10}$ and $\lambda = 2.82 \times 10^{10}$. The amplitude of the sine perturbation is around 10% in ρ, μ and λ. Figure 3.6 shows the density on the plane $z = 1000$ at the exact minimum.

The inversion algorithm starts from the constant material as initial guess, and iterates to find the sinusodial material variation. As an illustration of the convergence process, the density in the plane $z = 1000$, with the same contour levels are plotted in Fig. 3.7.

Just as in the case with one material grid point, the Lamé parameter λ shows the largest variation during the iteration process. The evolution of λ

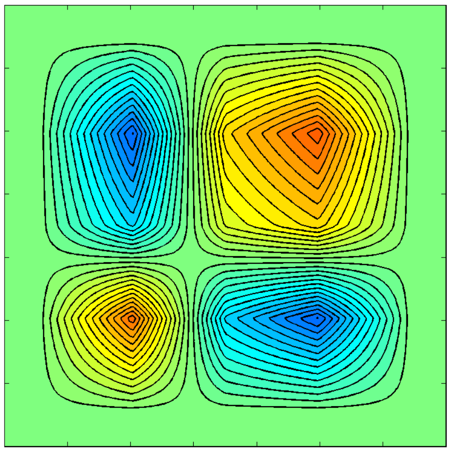


Figure 3.6: Density on the plane $z = 1000$ at the exact minimum.

is displayed in Fig. 3.8, showing it out of range of the plotting contour levels after 10, 20, and 40 iterations.

The convergence in Fig. 3.9, shows that around 200 iterations are needed to drive down the misfit to the level of round-off. However, Figs. 3.7 and 3.8 show that 50–60 iterations give a satisfactory picture of the material, corresponding do a misfit reduction of 4–5 orders of magnitude.

The input file for this example is given by

```
grid h=200 x=35000 y=35000 z=19500
time t=9 utcstart=09/10/2013:23:5:53.000
fileio verbose=1 path=run8 obspath=obs temppath=/tmp/bjorn pfs=1
supergrid gp=13 dc=.015
#
block r=2650 vs=2677.6503357897 vp=4998.11285141419
mparcart nx=3 ny=3 nz=2 init=0
#
mrun task=minvert mcheck=on
lbfgs nvectors=35 maxit=300 tolerance=1e-12 linesearch=on ihess0=scale-factors
mscalefactors rho=2650 mu=1.19e10 lambda=5.10e11
#
```

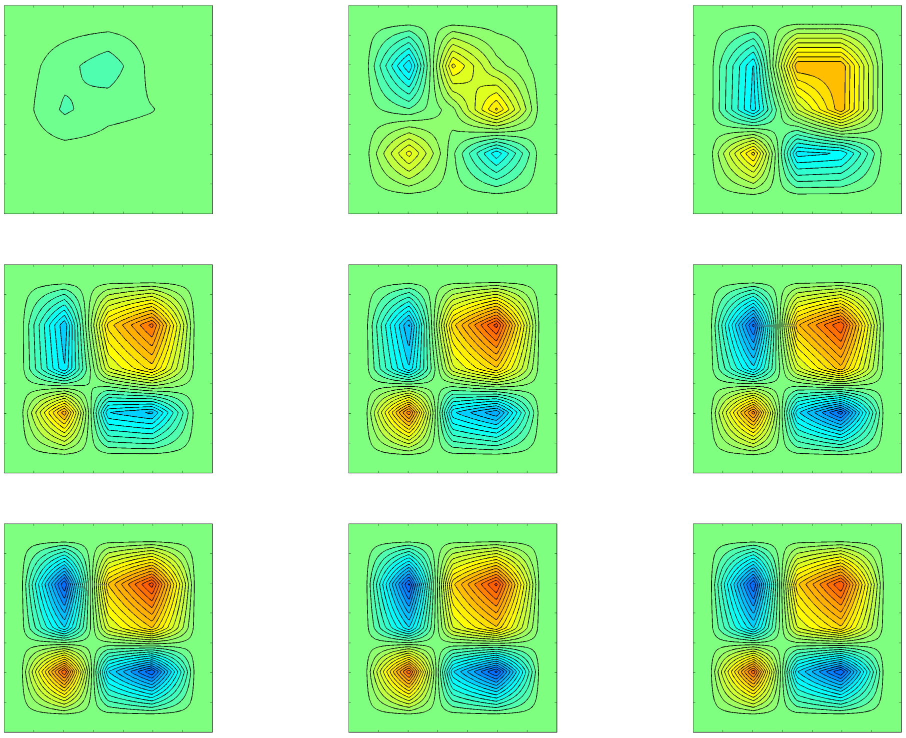


Figure 3.7: Density on the plane $z = 1000$ after 1, 10, 20, 30, 40, 50, 60, 70, and 80 iterations with the L-BFGS method.

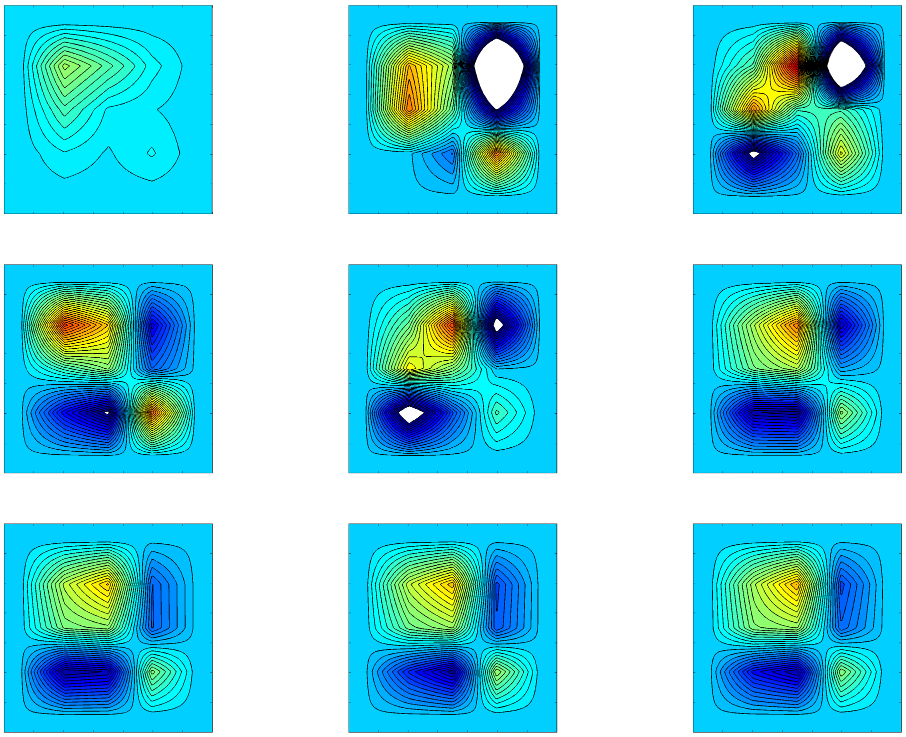


Figure 3.8: Lamé parameter λ on the plane $z = 1000$ after 1, 10, 20, 30, 40, 50, 60, 70, and 80 iterations with the L-BFGS method.

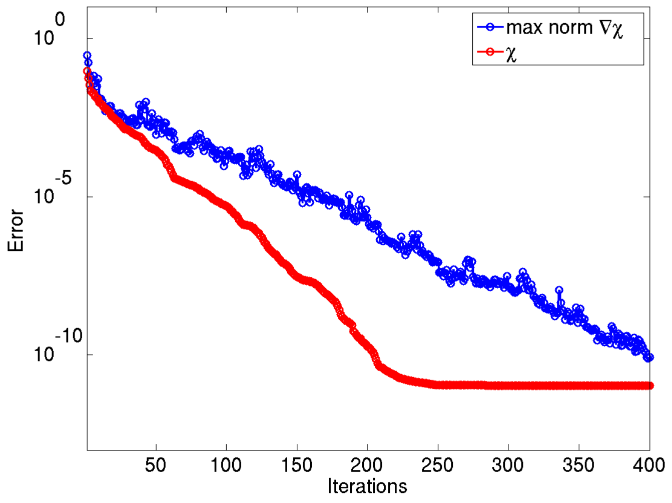


Figure 3.9: Convergence of misfit (red) and maximum norm of the gradient of the misfit (blue) for the computation shown in Fig. 3.7.

```
source x=17500 y=17500 z=2000 Mxy=1 m0=1e18 t0=1.5 freq=4 type=Gaussian
#
developer cfl=1.1
#
observation x=11500 y=11500 z=0 file=sta11
observation x=14500 y=11500 z=0 file=sta21
observation x=17500 y=11500 z=0 file=sta31
 ....
```

The first four lines set up the domain and discretization size. It uses the same syntax as the forward solver, *SW4*. The only additional items for the inverse problem are the two new paths, `obspath` and `temppath`, given at the `fileio` command. The `obspath` is the directory containing the observed seismograms. These data are read by the solver, nothing is output into `obspath`.

The `temppath` is a directory where temporary files associated with the outflow boundaries are stored. In order to reconstruct the forward solution during the backward solve, the forward solution on the outflow boundaries needs to be saved. Each processor that holds a part of the outflow boundary, writes its own file to `temppath`. These files can become fairly large, and there can be a lot of them. The read and write speeds to `temppath` will have a major effect on the preformance of the inverse solver. Preferably, `temppath` should be a directory on a disk that is local to the computational node. On the LC sytems, the directory `/tmp/username` is usually a good choice, if the files are not too large. The temporary files are deleted by *SW4mopt* when they are no longer needed, however if the program crashes, files might be left in `temppath`.

The next two lines,

```
block r=2650 vs=2677.6503357897 vp=4998.11285141419
mparcart nx=3 ny=3 nz=2 init=0
```

specify the material repesentation and the initial guess material. The computation starts from a zero perturbation on the constant reference material described by the `block` command. The material grid has $3 \times 3 \times 2$ points.

The L-BFGS method is specified by

```
lbfgs nvectors=35 maxit=300 tolerance=1e-12 linesearch=on ihess0=scale-factors
```

A maximum of 300 iterations are taken, using 35 L-BFGS vectors. The initial inverse Hessian will be computed from the scale factors. These are specified on the `mscalefactors` line below. The scale factors are also used to compute the step length in the minimization algorithm.

The source is given below the `mscalefactors` line. The source is specified with exactly the same syntax as in the forward solver. Below the `source` command, the CFL-number is set to 1.1. In *SW4mopt*, the maximum allowed CFL-number is hard coded to 1.3. By computing with a somewhat lower CFL-number, we allow the material wave speeds to increase by (1.3-1.1)/1.1 = 18% during the minimization. *SW4mopt* never changes the initially computed time step. However, the line search algorithm restricts the step size of the minimizer so that the maximum CFL-limit 1.3 is always respected. If the material makes the CFL-number getting too close to 1.3, the minimization step length will go to zero, and the minimizer fails. In that case, the computation has to be restarted with a smaller CFL-number.

The synthetic seismograms were prepared by the forward solver, run in the perturbed material. The seismograms are output in USGS-format, on files `sta11.txt`, `sta21.txt`, etc. up to `sta55.txt`. The synthetics are read into *SW4mopt* by the commands

```
observation x=11500 y=11500 z=0 file=sta11
observation x=14500 y=11500 z=0 file=sta21
observation x=17500 y=11500 z=0 file=sta31
....
```

etc. for all 25 stations. The synthetics were time stamped by the forward solver. The UTC time is registered in the header of the output time series on USGS-format. In this example, the UTC time was September 10th 2013 at 23:5:53. It is important that we specify the same start time for the inverse solver. This is done by the command

```
time t=9 utcstart=09/10/2013:23:5:53.000
```

in the input file.

Chapter 4

Keywords in the input file

The syntax of the input file is the same as in the forward solver, *SW4*. The input file consists of a number of lines with statements

```
command1 parameter1=value1 parameter2=value2 ... parameterN=valueN
# comments are disregarded
command2 parameter1=value1 parameter2=value2 ... parameterM=valueM
...
```

Each command starts at the beginning of the line and ends at the end of the same line. Blank and comment lines are disregarded. A comment is a line starting with a # character. The order of the parameters within each command makes no difference.

Parameter values are either integers (-2,0,5,...), real numbers (20.5, -0.05, 3.4e4), or strings (earthquake, my-favorite-simulation). Note that there must be no spaces around the = signs and strings are given without quotation marks and must not contain spaces.

A breif description of all commands is given in the following sections. The commands marked as [required] must be present in all *SW4mopt* input files, while those marked as [optional] are just that.

4.1 Material optimizer

4.1.1 The mrun command

Syntax:

```
mrun task=...  mcheck=...  tsoutput=...
```
Required parameters:
None

The `mrun` command specifies which task to perform. The possible tasks are given in the table below.

possible values of the task option	
Option	**Description**
minvert	Perform material inversion (default)
gradtest	Test gradient computation vs. a numerical derivative
hesstest	Compute the Hessian numerically
func1d	Compute and output a one dimensional cut of the misfit
func2d	Compute and output a two dimensional surface of the misfit
forward	Run one forward solve and exit.
minvert+src11	Perform material and source inversion, 11 parameter source.
minvert+src10	Perform material and source inversion, 10 parameter source.
minvert+src9	Perform material and source inversion, 9 parameter source.
minvert+src6	Perform material and source inversion, 6 parameter source.

Further specification of the `func1d`, `func2d` tasks, e.g., which parameter to vary, can be done with the `msurf` command. The generated cut/surface is output on a file named `fsurf.bin`, the format of which is described in Section **??**.

The intended use of `task=forward` is to generate synthetic seismograms for testing the material inversion.

`mcheck=on` tells the optimizer to check the computed material for reasonableness, e.g., that the density and μ are positive, after each iteration. `tsoutput=on` causes the time series (synthetic seismograms) to be output after each iteration. `mcheck` and `tsoutput` are only effective when the task is `minvert`.

mrun command parameters			
Option	**Description**	**Type**	**Default**
task	task to perform	string	minvert
mcheck	material checking (on or off)	string	off
tsoutput	output time series (on or off)	string	off

Currently, mcheck only outputs diagnostic messages, no attempt to correct the material if it is out of range is made.

4.1.2 The lbfgs command

Syntax:
`lbfgs nvectors=... ihess0=... maxit=... tolerance=... linesearch=...`
Required parameters:
None

Configure the L-BFGS method for minimizing the misfit. L-BFGS will iterate until `maxit` iterations are reached, or until the maximum norm of the scaled gradient of the misfit is less than `tolerance`.

The option `linesearch=off` switches off the line search step of L-BFGS. This is usually not stable. The default `linesearch=on` switches on a standard line search algoritm. However, L-BFGS is only guaranteed to be stable if the so called Wolfe condition is satisfied. The option `linesearch=wolfe` switches on the line search with additional logic to satisfy the Wolfe condition. Line search with the Wolfe condition is computationally expensive, since the gradient of the misfit has to be evaluated at least once. Therefore, it is advisable to try the standard line search first, we have found it to work satisfactory in many cases.

L-BFGS builds and approximation of the inverse Hessian, represented by `nvectors` vectors, which can be thought of as a rank-`nvectors` approximation. The convergence rate is usually better for larger values of `nvectors`. The method requires an initial guess for the inverse Hessian. The option `ihess0=scale-factors` uses an initial guess based on the scale factors of the problem, while `ihess0=gamma` uses an initial guess by an estimate of the size of the Hessian along the search directions, given by formula (7.20) in [1].

lbfgs command parameters			
Option	**Description**	**Type**	**Default**
nvectors	Number of l-bfgs vectors to keep	int	10
ihess0	Initial guess for inverse Hessian	string	gamma
maxit	Maximum number of iterations	int	10
tolerance	Termination criterion for gradient	float	10^{-12}
linesearch	Line search method (on, off, or wolfe)	string	on

4.1.3 The nlcg command

Syntax:
`nlcg maxit=... tolerance=... linesearch=... subtype=... maxsubit=...`
Required parameters:
None

Configure the non-linear conjugate gradient (NLCG) method for minimizing the misfit. NLCG will iterate until `maxit` restarts are reached, or until the maximum norm of the scaled gradient of the misfit is less than `tolerance`. CG methods are usually restarted every nth iteration, where n is the number of unknowns. The behavior of the iterations is controlled by the two parameters `maxit` and `maxsubit`. `maxit` gives the maximum number of restarts, and `maxsubit` is the number of iterations between the restarts. By default `maxsubit` is set to n.

The option `linesearch=off` switches off the line search step in NLCG. The default `linesearch=on` switches on a standard line search algoritm. The initial step size is computed by approximating the minimization functional by a quadratic surface.

The `subtype` option makes it possible to specify the Fletcher-Reeves or the Polak-Ribière variants of the NLCG. The Polak-Ribière algorithm forces a restart more often than Fletcher-Reeves. With `subtype=polak-ribiere`, `maxit` needs to be set large enough to allow for the additional restarts.

nlcg command parameters			
Option	**Description**	**Type**	**Default**
maxit	Maximum number of iterations	int	10
tolerance	Termination criterion for gradient	float	10^{-12}
linesearch	Line search method (on or off)	string	on
subtype	fletcher-reeves or polak-ribiere	string	polak-ribiere
maxsubit	Number of subiterations in CG	int	# unknowns

4.1.4 The mscalefactors command

Syntax:
`mscalefactors rho=... mu=... lambda=... file=... misfit=...`
Required parameters:
None

Introducing scale factors will improve the convergence rate of the minmizer, by reducing the condition number of the Hessian of the misfit. For quadratic problems, the scale factors form a diagonal pre-conditioning matrix. There is one scale factor for each unknown. Ideally, the scale factor for the ith unknown, x_i, should be $1/\sqrt{H_{ii}}$, where H_{ii} is the ith diagonal element of the Hessian. The `file=` option specifies a file name, from which the scale factors are read. The number of scale factors on the file should be equal to the number unknown parameters. The format of the file is described in Section **??**.

If the Hessian is not known, an alternative is to set the scale factors to reference sizes of the parameters. If the material parameterization is made such that each parameter is identifiable with one of the material properties, ρ, μ, or λ, the options `rho=`, `mu=`, and `lambda=` can be used to specify three different scales. These scale factors are used throughout for all unknowns of the respective type.

The `misfit=` is a factor that modifies the input scale factors. It is based on the observation that the scale of $1/\sqrt{H_{ii}}$, with $H = \partial^2 f/(\partial x_i^2)$, is actually $x_r/\sqrt{f_r}$, where x_r is the scale of the unknown x_i, and f_r is the scale of the objective function. All input scale factors will be divided by the square root of the value of the `misfit` parameter. The condition number of the Hessian is not affected by multiplying all factors by a constant, but the multiplier will affect the maximum step length, and is currently needed to sometimes reduce the allowed step size. This is a temporary measure that should be removed, once the line search algorithm has been improved.

mscalefactors command parameters			
Option	**Description**	**Type**	**Default**
rho	Density scale factor	float	1
mu	Scale of Lamé parameter μ	float	1
lambda	Scale of Lamé parameter λ	float	1
file	Name of file containing scale factors	string	None
misfit	Multiplier for scale factors	float	1

4.1.5 The mfsurf command

Syntax:
`mfsurf var=... i=... j=... k=... pmin=... pmax=... npts=... var2=... i2=... j2=... k2=... pmin2=... pmax2=... npts2=...`
Required parameters:
None

The command `mfsurf` controls the selections for `mrun task=func1d` and `mrun task=func2d`. The command assumes that the material parameters can be interpreted as values of ρ, μ, or λ on a logically rectangular grid, for example, when using material parameterization by the `mparcart` command.

`var=`, `i=`, `j=`, `k=` specify one parameter. For example the density at the point with index $(3, 2, 4)$ in the coarse material grid defined by `mparcart`, is selected by

```
mfsurf var=rho i=3 j=2 k=4
```

The command

```
mfsurf var=rho i=3 j=2 k=4 npts=30 pmin=-250 pmax=250
```

specifies the misfit as function of this parameter on the interval [-250,250], discretized by 30 points. To compute and save the specified function on a file, run *SW4mopt* with `mrun task=func1d`.

The second set of input variables, `var2=`, `i2=`, etc. are used for the second dimension when a two dimensional misfit surface is specified with `mrun task=func2d`.

mfsurf command parameters			
Option	**Description**	**Type**	**Default**
var	variable (rho, mu, or lambda)	string	rho
i	i-index	int	1
j	j-index	int	1
k	k-index	int	1
pmin	lower parameter limit	float	-300
pmax	upper parameter limit	float	300
npts	Number of discretization points	int	10
var2	variable (rho, mu, or lambda)	string	rho
i2	i-index	int	1
j2	j-index	int	1
k2	k-index	int	2
pmin2	lower parameter limit	float	-300
pmax2	upper parameter limit	float	300
npts2	Number of discretization points	int	10

4.2 Material parameterization [required]

Several ways to parameterize the material will be tried. Currently, only parameterization through a coarser grid is possible, by the `mparcart` command.

4.2.1 mparcart

Syntax:

```
mparcart nx=...  ny=...  nz=...  init=...
```

Required parameters:

```
nx, ny, nz, init
```

The command mparcart defines the material by interpolation from a coarse grid. The coarse grid stores the offsets in ρ, μ and λ from a reference material. The init parameter is either 0, meaning that all offsets are initialized to zero, or the name of a file with previously computed offsets. If a file name is specified, the material is initialized with the values stored on the file. The material optimizer stores the current values of the offsets on the file `parameters.bin` after each iteration. Hence, a previous computation can be restarted by specifying `init=parameters.bin`.

mparcart command parameters			
Option	**Description**	**Type**	**Default**
nx	Number of points in x	int	none
ny	Number of points in y	int	none
nz	Number of points in z	int	none
init	initial guess, file name or 0	string	none

Currently, the complete material grid is stored in each processor. For very large problem sizes, the amount of memory can be a limitation.

4.3 Output

4.3.1 mimage

Syntax:
```
mimage x=...  y=...  z=...  cycle=...  cycleInterval=...
file=...  mode=...  precision=...
```
Required parameters:
Location of the image plane (x, y, or z)
Time for output (cycle, or cycleInterval)

Material images are similar to the image command of *SW4*. The main difference is that `mimage` defines image output related to the iterations of the minimization algorithm. In the `cycle` and `cycleInterval` options, one cycle is interpreted as one iteration of the minimization algorithm. The options `x`, `y`, `z`, `file`, `mode`, and `precision` are identical to the options with the same names in the `image` command. `mimage` only outputs images of material properties. The supported modes are given in the table below.

mimage mode options	
Value	**Description**
rho	Density
lambda	1st Lamé parameter
mu	2nd Lamé parameter (shear modulus)
p	Compressional wave speed
s	Shear wave speed
gradrho	Gradient of misfit w.r.t. density
gradlambda	Gradient of misfit w.r.t. 1st Lamé parameter
gradmu	Gradient of misfit w.r.t. 2nd Lamé parameter
gradp	Gradient of misfit w.r.t. Compressional wave speed
grads	Gradient of misfit w.r.t. Shear wave speed

Note, the misfit gradients are computed with respect to the material properties at each grid point. When using a material parameterization, the misfit with respect to the parameters is given by the chain rule as a combination of these gradients with the derivative of the parameterization.

mimage command parameters			
Option	**Description**	**Type**	**Default**
cycle	Minimizer cycle to output image (≥ 0)	int	0
cycleInterval	Minimizer cycle interval to output a series of images (≥ 1)	int	1
file	File name header of image	string	mimage
precision	Floating point precision for saving data (float or double)	string	float
mode	The field to be saved	string	rho

Bibliography

[1] J. Nocecdal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.