SAGA User Manual 5.0: An inversion software package

November 24, 2003

Peter Gerstoft
Marine Physical Laboratory
Scripps Institution of Oceanography
University of California at San Diego
Email: gerstoft@ucsd.edu

Executive summary: The use of full field inversion methods or matched field processing for target localization has been shown to be effective for determination of target bearing, range, and depth in deep and shallow water. The performance of these methods is strongly dependent on accurate information about the environmental parameters. Previously, the lack of this knowledge inhibited the application of these methods in shallow water.

The objective of this report is to document a general software package SAGA (Seismo Acoustic inversion using Genetic Algorithms) which has been developed for assisting in estimating environmental parameters. The approach taken here is data-independent and therefore the SAGA inversion code has been applied to many types of data: single and multi frequency data on a vertical array, coherent and incoherent transmission loss, reverberation data, reflection coefficients from the sea bottom, and tropospheric electromagnetic data.

Global optimization using a directed Monte Carlo (random) search based on genetic algorithms is used to identify representative set of parameters. Genetic algorithms are based on an analogy with biological evolution, one of the most efficient optimization systems. Both the maximum likelihood estimate as well as uncertainty estimates are provided. The results are presented using a Bayesian framework. This approach is also applicable to data fusion, i.e. when combining information from several sensors.

Abstract: The purpose of many experiments is to identify environmental parameters which, when used as input to a forward model, accurately model the observed data. It is also useful to have some indication of the uncertainty of the estimated parameters. SAGA is a software package that helps the user determine the best set of parameters to match a given data set. At present, the package consist of nine modules, one for each forward model. The forward codes used are:
OAST (wavenumber integration transmission loss model),
OASR (wavenumber integration reflection coefficients model),
SNAP (normal modes),
SNAPRD (adiabatic normal modes),
POPP (normal mode reverberation model),
PROSIM (broadband adiabatic normal modes),
CPROSIM (broadband adiabatic normal modes),
ORCA (broadband adiabatic normal modes),
GAMARAY (broadband ray theory),
RAMGEO (Range dependent Parabolic Equation),
and TPEM (tropospheric parabolic equation).
Several types of observed data can be used in these inversions: single or multi-frequency pressure on a vertical array, coherent or incoherent transmission loss, reflection coefficients, reverberation data, or tropospheric electromagnetic data. For any parameter estimation problem the issue of error assessment must be addressed. In SAGA it is addressed by estimating *a posteriori* distributions.

It appears that there has been a natural evolution in search methods since the search engine for SAGA was developed. If an update is needed. the modular structure of SAGA should make it an easy task However, often the search engine is a low priority relative to the other problems the Ocean-Acoustician faces.

The word SAGA stems from Icelandic and simply means a story. By a saga is usually understood a prosaic long story from the middle ages. It tends to go on forever. Scientifically it stands for Seismo-Acoustic inversion using Genetic Algorithms.

New features in version 3.0:

- Several errors have been corrected in both manual and code.
- The forward codes PROSIM and TPEM have been much improved.
- New objective functions are introduced.
- It is possible to compute contour plots of Cramer-Rao bounds.
- SNAP does not recompute the modes if searching only for geometric parameters. For these cases the CPU-time is reduced by about 95 %.
- 3D Array localization capability when using SNAP.
- Arrays can be any geometry.

New features in version 4.0:

- The forward model CPROSIM (broadband adiabatic normal mode) has been included. CPROSIM finds the propagating and non-propagating normal modes. Thereby also the near field is accurately calculated.
- Line plots of sensitivity.
- for the optimal environment POST writes a transfer-function file. This is done for the forward models SNAP, PROSIM and CPROSIM.

New features in version 4.1:

- The forward model RAMGEO (Range dependent Parabolic Equation) has been included.
- A table of all used forward models is stored to make sure that no model is computed more than once. For small optimization problems this can improve performance.
- All used forward models is written to disk for the post processing.
- a scatter plot of value of objective function for each parameter can be displayed.
- During the inversion the posteriori analysis can continuously be updated.
- The TPEM forward model has several options for handling range-dependent trilinear profiles and doing refractivity-from-clutter inversions.

New features in version 5:

- The forward model GAMARAY has been included.
- The forward model ORCA90 has been included. ORCA90 seems very computationally efficient in real axis mode.
- Gibbs sampling of the likelihood functions.
- Enumerative sampling of the likelihood functions.
- For all forward models it is possible to do a local optimization using the Powell method.
- At the end of each run a local search using the Powell method is carried out.
- it is possible to use random seeds to initialize the search algorithms.
- Several options to support the inversion of atmospheric refractivity profiles based on clutter return in radar images has been implemented.

Keywords: genetic algorithms ◦ inversion ◦ optimization ◦ SAGA

Contents

1
Introduction

SAGA, Fig. 1, is a computer code for inversion of observed data. Its purpose can best be understood by dividing the inversion process into five parts:

(I) Discretization of the environment and discretization or transformation of the data.

(II) Efficient and accurate forward modeling.

(III) A suitable objective function.

(IV) Efficient optimization procedures.

(V) Uncertainty analysis.

Item (I) is concerned with how to collect and discretize a wave field in order to have the necessary information for the inversion, and to determine the parameters for which inversion is feasible. Item (I) leads to a set of known environmental parameters and *a priori* bounds for the unknown parameters. Based on the above parameters, a replica field can be computed by a forward acoustic model, item (II). The observed data and the replica are then compared though an objective function, item (III). Through an iterative scheme, item (IV), the match between the observed and computed data is maximized by varying the environmental parameters. From the best models obtained, it is possible to provide estimates of the value of the parameters and their uncertainty and importance, item (V). The best solution is not very interesting without a proper statistical analysis of the result.

Complete inversion requires equal attention to all five items, as illustrated in Fig. 2. It is also clear that each item depends on its predecessor. Therefore it is natural that earlier research has focused on the first two items. The present code is concerned with the solution of items (III), (IV) and (V).

The forward models, item (II), presently used are: SNAP (normal modes) and SNAPRD (adiabatic normal modes) [1], OAST (wavenumber integration transmission loss model) and OASR (wavenumber integration reflection coefficients model) [2, 3],

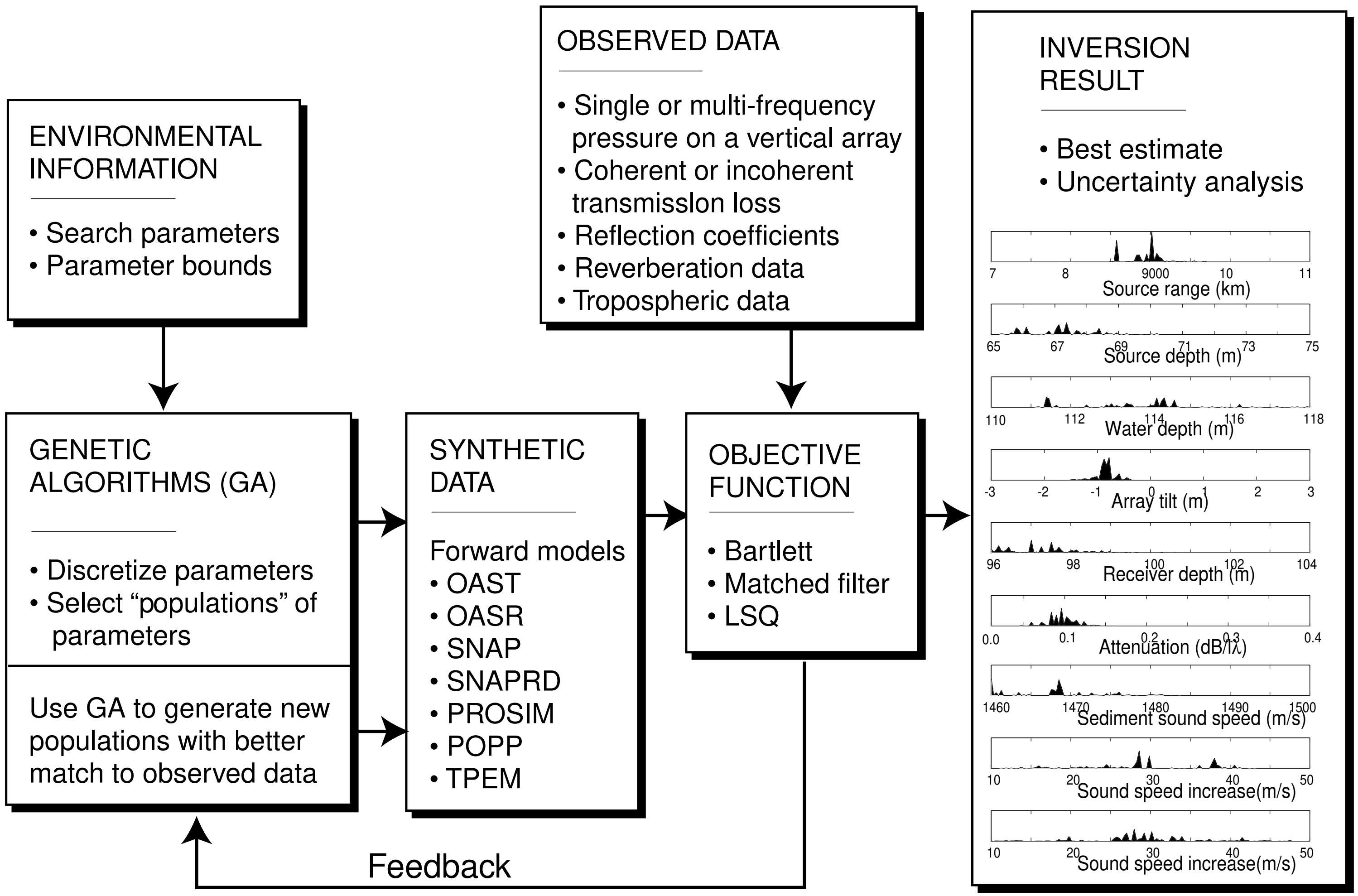

Figure 1 *Flow diagram for the SAGA code.*

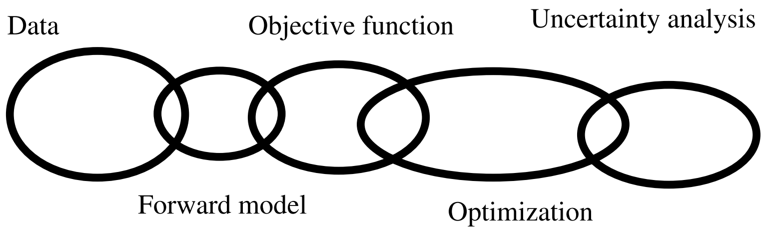

Figure 2 *The weakest link determines the outcome of an inversion procedure.*

POPP (normal mode reverberation model) [4], PROSIM, CPROSIM (broadband adiabatic normal modes) [5, 6, 7], RAMGEO (parabolic equation method) [12, 13] and TPEM (tropospheric parabolic equation) [14, 15].

The optimization, item (III), is based mainly on genetic algorithms (GA) [16], but also on Gauss-Newton (GN), a hybrid combination of GA and GN as described in [16, 18] and simulated annealing (SA). To ensure convergence to the global optimum, we use several populations in parallel for the genetic algorithms. The search efficiency can be increased by combining the GA and GN methods. This task is handled by the optimization module SAGA.

Two versions of SA are available: Fast SA (FSA) as described in [19] and Very Fast SA (VFSA) as described in [20, 21]. The simulated annealing routines are provided so that the user can assess the estimates obtained using different optimization methods. The SA methods do not provide an estimate of *a posteriori* distributions.

Analysis of the solution, item (IV), is achieved by examining *a posteriori* distributions with the post-processor module POST. The solution can also be assessed by local methods such as singular-value decomposition [16, 22, 23], or by plotting the ambiguity surface.

From the above description it is clear that inversion is a complex procedure and in order to be successful many aspects have to be considered: Forward modeling [24], signal processing [25, 26], optimization and estimation theory [27], global optimization [31], deterministic discrete inversion [28], deterministic continuous inversion [29], and stochastic inversion [30].

2

Background

The non-linear inverse problem is stated as an optimization problem: Find the model vector $\mathbf{m}$, or parameter set, that minimizes the objective function $\phi = f(\mathbf{p}, \mathbf{q}(\mathbf{m}))$, where $\mathbf{p}$ is the observed data and $\mathbf{q}$ is the synthetically generated data using a given forward model with a set of physical parameters $\mathbf{m}$. Normally, in genetic algorithms the objective function is maximized, but here it is minimized, in accordance with the precept of simulated annealing.

Global optimization methods accept that the objective function is irregular and attempt to find the global minimum, without an exhaustive search. An advantage of global optimization is that it requires only the value of the objective function at arbitrary points in space. The problem can then be solved without further knowledge of the objective function. Thus, once the global inversion method has been tuned, any forward model can be used. Early solutions to the global problem were attempted using a simple Monte Carlo method, whereas modern methods use directional searches such as *genetic algorithms* (GA) or *simulated annealing* (SA). The SA method has been applied to ocean acoustics in [19, 32, 33, 34]. A comparison of SA and GA from an ocean acoustics viewpoint has been presented in [35].

2.1 Optimization using Genetic Algorithms

Genetic algorithms are analogous to biological evolution. They have been applied to seismics [36, 37, 38, 39], and ocean acoustics [16, 40, 41, 42, 43]. The basic principle of GA is simple: From all possible model vectors, an initial population of q members is selected. The fit of each member is computed based on the fit between the observed data and the computed data. Then through a set of evolutionary steps, the initial population evolves in order to become more fit. An evolutionary step consists of selecting a parental distribution from the population based on the individual's fit. The parents are then combined in pairs and operators are applied to them to form a set of children. Traditionally the crossover rate and mutation rate operators have been used. Finally, the children replace part of the population to increase the match between observed and synthetic data.

The environment is discretized into M parameters in a model vector $\mathbf{m}$. Each

	*	*	*	*	*	*	*	*	
$i_j =$	0	0	0	0	0	0	0	0	m_j^{min}
$i_j =$	0	0	0	0	0	0	0	1	$m_j^{min} + 1\,\Delta m$
$i_j =$	0	0	0	0	0	0	1	0	$m_j^{min} + 2\,\Delta m$
$i_j =$	0	0	0	0	0	0	1	1	$m_j^{min} + 3\,\Delta m$
⋮									⋮
$i_j =$	1	1	1	1	1	1	1	1	m_j^{max}

Figure 3 *Binary coding of model parameters.*

parameter m_j, $j = 1, \dots M$, can assume 2^{n_j} discrete values according to an *a priori* probability distribution (Gaussian, rectangular or based on *a priori* information). Here a rectangular distribution between a lower and upper bound $[m_j^{\min}, m_j^{\max}]$ is used. We have, see Fig. 3,

$$m_j = m_j^{\min} + i_j\,\Delta m_j\ , \quad i_j = 0, \dots, 2^{n_j} - 1\ , \tag{1}$$

where

$$\Delta m_j = \frac{m_j^{\max} - m_j^{\min}}{2^{n_j} - 1}. \tag{2}$$

A major difference between simulated annealing and GA is that GA uses q model vectors at the same time, where q is the population size, while simulated annealing only uses one. GA consists essentially of three operators: selection, crossover and mutation.

<u>Selection:</u> In order to establish a new population, also with q members, $f\,q$ parents must be selected, $0 < f < 1$. The choice is made with a probability proportional to the fit of its members. The simplest probability is given by

$$p_k = \frac{1 - \phi(\mathbf{m}_k)}{\sum_{l=1}^{q}[1 - \phi(\mathbf{m}_l)]}, \quad k = 1, \dots q\ . \tag{3}$$

The introduction of a temperature, as in simulated annealing, gives us the opportunity to stretch the probability and improve the algorithm performance [38]. Indeed, at the first stage of the procedure, by stretching the fit we avoid to choose as parents only the members with the better fit, which would otherwise tend to dominate the population; later in the optimization, this stretching leads to a better discrimination between models with very close fit. In order to take into account this new parameter, the probability is rewritten as

$$p_k = \frac{\exp\left[-\phi(\mathbf{m}^k)/T\right]}{\sum_{l=1}^{q}\exp\left[-\phi(\mathbf{m}^l)/T\right]}, \quad k = 1, \dots q\ . \tag{4}$$

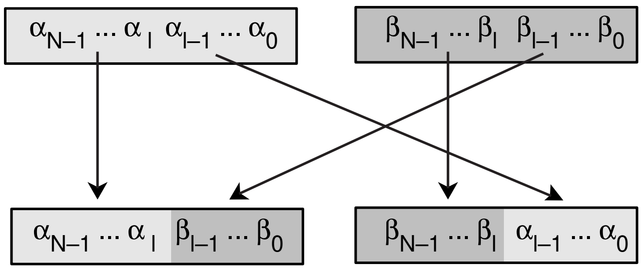

Figure 4 *Crossover is a binary exchange of l bits between the binary codes for two model parameters. l is chosen randomly.*

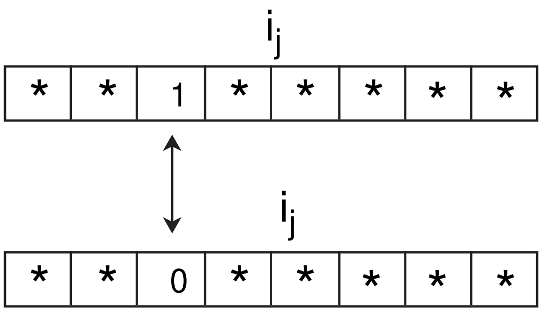

Figure 5 *Mutation is a random change of one bit.*

But, as with simulated annealing, the choice of the temperature T is difficult. It must be neither too high nor too low. A good compromise is a temperature of the same magnitude as the objective function, here $T = \min[\phi(\mathbf{m}^k)]$. During optimization, the fit increases and temperature decreases.

Crossover For each set of parents, each consisting of a model vector, two children are constructed, and for each parameter in the model vector, each child may either be a direct copy of one parent, with probability $1 - p_{\mathrm{x}}$, or it can be a bit crossover of the two parents with crossover probability p_{x}, see Fig. 4. The crossover point is chosen randomly in the interval $[1, N-1]$, where N is the number of bits used in the coding. Different techniques are available to perform this crossover of the population, e.g., single point crossover where the entire "chromosome" is used once, and the multiple point crossover where the chromosome is divided into genes related to each parameter on which the crossover is applied. Multiple point crossover is used here.

Mutation: This is a random change of one bit in the model vector, with probability p_m in order to better explore the search space (see Fig. 5).

It is possible that a run of a GA will approach a local minimum. In order to increase the probability of finding the global minimum, several independent populations M_{par} are started. This is also advantageous for collecting statistical information in order to estimate the probabilities, as shown in the examples in the next section.

In the present implementation there are relatively few GA parameters which in principle have to be tuned for each application, and the precise value of each of these seems to be not that important. We usually only vary M_{par} in order to adjust the execution time to the available CPU time. Based on our experience, the following values are recommended:

- The population size q should be large enough to allow the model vectors to represent several minima, but also small enough to allow several iterations to be performed; $q = 64$ seems to be a good compromise.
- The reproduction size f should be large enough to allow the fittest individuals to stay in the population during the iterations; f should be less than 0.9, $f = 0.5$ is recommended.
- The crossover rate depends on how independent the parameters in the model vector are. A crossover rate p_{x} close to 1.0 seems to be a good choice for independent parameters; for dependent parameters a lower value, e.g. $p_{\mathrm{x}} = 0.8$, is recommended.
- It has been found that a high mutation rate gives the best result, $p_{\mathrm{m}} = 0.05$ is recommended.
- The number of forward computations N_{forw} for each population should be relatively low ($N_{\mathrm{forw}} = 1000$–5000).
- The number of parallel populations N_{pop} depends on the application. To obtain a reasonable estimate of the inversion parameters, $N_{\mathrm{pop}} = 1$ is sufficient. For computing the probability distribution it must be larger, e.g. $N_{\mathrm{pop}} = 50$.

2.2 *A posteriori* statistics

Before measurement, the information about the models is reflected in the *a priori* distribution $\rho(\mathbf{m})$, and after the experiment, the information about the models is reflected in the *a posteriori* distribution $\sigma(\mathbf{m})$. These distributions are related through the likelihood function $\mathcal{L}(\mathbf{m})$, which is a measure of the goodness of fit between the observed data and the data generated using a physical model and the environment $\mathbf{m}$ (Bayes Theorem):

$$\sigma(\mathbf{m}) = \mathcal{L}(\mathbf{m})\rho(\mathbf{m}) \; . \tag{5}$$

When maximizing $\sigma(\mathbf{m})$ the *Maximum A Posteriori* (MAP) estimate of the parameters is obtained and when maximizing $\mathcal{L}(\mathbf{m})$ a *Maximum Likelihood* (ML) estimate of the parameters is obtained.

Due to multi-dimensionality, the *a posteriori* distribution, often with $M > 10$, is not suitable for graphic display, and therefore mainly integral properties of this distribution are of interest. From the *a posteriori* probability distribution, information will be extracted to describe the solution. The following quantities are of interest: The MAP solution $\hat{\mathbf{m}}^{\mathrm{MAP}}$

$$\hat{\mathbf{m}}^{\mathrm{MAP}} \equiv \arg \max_{\mathbf{m}\in\mathcal{M}} \sigma(\mathbf{m}) \ , \tag{6}$$

the expectation $\mathsf{E}_\sigma[\mathbf{m}]$

$$\mathsf{E}_\sigma[\mathbf{m}] \equiv \int_{\mathcal{M}} \mathbf{m}\sigma(\mathbf{m})\,\mathrm{d}\mathbf{m} \ , \tag{7}$$

where $\mathrm{d}\mathbf{m} = \mathrm{d}m^1 \ldots \mathrm{d}m^M$. The covariance matrix $\mathsf{Cov}_\sigma[\mathbf{m}]$

$$\mathsf{Cov}_\sigma[\mathbf{m}] \equiv \mathsf{E}_\sigma \left\{ [\mathbf{m} - \mathsf{E}_\sigma(\mathbf{m})][\mathbf{m} - \mathsf{E}_\sigma(\mathbf{m})]^{\mathrm{T}} \right\} \ , \tag{8}$$

the 1-D marginal *a posteriori* probability densities $\sigma^i(m^i)$ for parameter m^i

$$\sigma^i(m^i) \equiv \int \sigma(\mathbf{m})\mathrm{d}m^1 \ldots \mathrm{d}m^{i-1}\,\mathrm{d}m^{i+1} \ldots \mathrm{d}m^M \ , \tag{9}$$

and higher-dimensional marginals are defined similarly to Eq. (9). The marginal distributions are the most important in interpreting the inversion result.

A problem with this approach is the derivation of a likelihood function [45]. The likelihood function depends on the stochastic model for the data, an example is given in Annex A. The standard approach in SAGA is to estimate empirically the likelihood function. Knowing that the likelihood function is usually related to the objective function $\phi(\mathbf{m})$ through an exponential $\mathcal{L} = \exp(-\phi(\mathbf{m})/\hat{\nu})$ [30], where $\hat{\nu}$ is the estimated noise power, the following scaling was used:

$$\mathcal{L}_{\mathrm{emp}}(\mathbf{m}) = \exp\left(-[\phi(\mathbf{m}) - \phi(\mathbf{m}_0)]/T\right) \ , \tag{10}$$

where ϕ is any objective function and $\mathbf{m}_0$ is the estimated parameter vector corresponding to the optimal value of the objective function. T is the "temperature". Experimentally, it was found that a good value for T is the average of the 50 best objective functions obtained during the optimization, minus the best value of the objective function. It should be noted that this value of T is not intended to estimate the noise, but rather to produce a reasonable value with which to estimate the uncertainties of the parameters. The advantage of this scheme is that it works irrespective of the stochastic model for the data or likelihood function.

During the optimization, all obtained samples of the search space are stored, and used to estimate the *a posteriori* probabilities using importance sampling [45].

For the N_{obs} observations, the *a posteriori* probability for the kth model vector is estimated by

$$\hat{\sigma}(\mathbf{m}_k) = \frac{\mathcal{L}(\mathbf{m}_k)\rho(\mathbf{m}_k)}{\sum_{j=1}^{N_{\text{obs}}} \mathcal{L}(\mathbf{m}_j)\rho(\mathbf{m}_j)} \ . \tag{11}$$

For the ith parameter m^i in the model vector the marginal probability distribution for obtaining the particular value κ can be found by summing Eq. (11):

$$\hat{\sigma}^i(\kappa) = \sum_{k=1}^{N_{\text{obs}}} \hat{\sigma}(\mathbf{m}_k)\ \delta(m_k^i - \kappa)\ , \tag{12}$$

where δ is the delta-function. In this particular implementation, several independent GA searches are started in parallel. It was found that when executing several parallel runs, it is sufficient to save the last model vectors in a population within each run [16].

3

Cook book for inversion

The following is a check list for doing inversions with SAGA:

1. Generate a baseline environmental model using all available information.
2. Decide on one or two appropriate forward models.
3. Perform a few forward runs based on the baseline model to identify the main propagation characteristics. For the inexperienced SAGA user it is probably advantageous to use a stand-alone version of the forward model.
4. Decide on type of observed data and SAGA data format.
5. Decide on objective function and scaling of observed data and replica.
6. Generate the data synthetically using the following procedure:
   (a) Specify option **W** in the input file.
   (b) Delete the *.in file.
   (c) Run SAGA. It will generate a *.obs file containing synthetic data based on the environment in the *.dat file. It will stop when the data has been generated because there is no observed data (*.in file).
   (d) Copy this file (*.obs) into the *.in file. SAGA always reads the observed data from the *.in file.
7. Make a short SAGA run using only a few hundred forward model calls, and invert only for a few environmental parameters.
8. Is the forward model sufficiently fast, i.e. a runtime about 1 second, or is it necessary to simplify the environment/forward model. Does it seem to produce reasonable results?
9. Genetic algorithms parallel evolution in that it is an indeterminate process. This is, however, impractical and the program stops when a certain number of forward solutions have been generated. The number of forward model runs is the product of the number of populations, N_{pop}, and number of forward model runs in a population, N_{forw}. Usually from 1,000–40,000 forward model

runs are carried out. The number of forward runs depends on available CPU-time, the information content in the data and the number of parameters to be estimated.

10. Generate a larger inversion example with more forward modeling runs and environmental parameters.

11. Use the post-processor program, POST, to analyze the results.

    POST displays the best, maximum, and mean parameter vectors. These are: the parameter vector that gave the best match between the observed and calculated data; the parameter vector that corresponds to the maximum of the *a posteriori* probability distribution; and the parameter vector that corresponds to the mean of the *a posteriori* distribution.

    In addition, the normalized standard deviation for each parameter i is computed, $\sigma_i/(F^i_{\max}-F^i_{\min})$. This gives an indication of the spread of a distribution for each parameter. As a limiting case for a flat distribution, we obtain a normalized standard deviation of $1/(2\sqrt{3}) = 0.29$. For a distribution that is flat in half of the search interval, we obtain $1/(4\sqrt{6}) = 0.10$.

12. Plot the results from POST. This is best done using MATLAB. This will show the marginal *a posteriori* distributions. If option **p** is specified, a plot of the observed and synthetic data for the best environment is also shown.

13. Check if some parameters are coupled, i.e. depend on each other. This can be investigated using one of the following approaches:

    (a) Stochastic approach. Plot the 2-D marginal *a posteriori* distribution between two parameters. The approach is similar to when the 1-D *a posteriori* distributions are plotted. First edit the SAGA option line in the *.dat file to include option **m45** to compute the 2-D marginal *a posteriori* distribution between, e.g., parameters 4 and 5, and then run POST.

    (b) Deterministic approach. Plot the ambiguity function between two parameters. The remaining parameters are kept at their nominal values. Edit the SAGA option line to include option **C**, and then run SAGA. Note, that it is then not required to run POST.

    The result of both approaches is a contour plot. This can be plotted using CPLOT, see Sect. 5.1.

14. In the event some of the parameters are coupled the inversion can be made more efficient by reparameterization using shape functions. This is option **E**, see also the examples in Sect. 9.

15. Are some parameters unimportant? It is not necessary to optimize unimportant parameters, or maybe they can be combined with other parameters to a

single significant parameter. In the latter case this can be accomplished by the use of shape functions. This is option **E**, see also the examples in Sect. 9.

16. When the synthetic inversion appears to make sense, it is time to start on the actual data collected at sea.

17. It is assumed here that good data are available. Better data result in better and more robust inversions. Some assistance in transforming the data into SAGA format is available from the MATLAB files.

18. Start observed data inversion with a short run to ensure that every thing works well.

19. There is no way to automatically parameterize an environment, and repeated trial and error is the suggested approach. In order to check the robustness of the discretization of the environment, several environments should be tried out.

20. That's almost it. If the results look good, congratulations! Otherwise go back to step 1 and try again.

4
Installing SAGA

4.1 SAGA directory

The SAGA home directory is defined on UNIX systems by the following command in your .login:

```
setenv SAGADIR your-SAGA-root-directory
```

This could e.g. be $HOME/saga. SAGA is able to run on multiple platform mounted under NFS. In order to do that we have host and compiler dependent variables, $HOSTTYPE that should be setup in your system and $FORTRAN that is a variable introduced to identify the Fortran compiler. For execution of SAGA you must include the scripts and executables in your search path, i.e. $SAGADIR/bin. The following line should be inserted in your .login:

```
set path = ($SAGADIR/bin $SAGADIR/bin/$HOSTTYPE-$FORTRAN $path)
```

In my Linux .login these are defined as (ifc is the Intel Fortran compiler; it can be downloaded free of charge for academic users)

```
setenv FORTRAN ifc
setenv HOSTNAME 'uname -n'
```

In my Sun Solaris HOSTNAME is system defined (to Solaris) and the Fortran is defined as

```
setenv FORTRAN Solaris
```

4.2 Loading SAGA files

For users running UNIX, the whole directory tree is provided in a compressed **tar** file with the name

```
saga.tar.Z
```

Place this file in your desired root directory $HOME and issue the commands:

`uncompress saga.tar.Z`

`tar xvf saga.tar`

which will install the directory tree:

$SAGADIR	SAGA root directory
$SAGADIR/src	Source files for SAGA
$SAGADIR/src/snap	Source files for SNAP range independent
$SAGADIR/src/snaprd	Source files for SNAPRD adiabatic modes
$SAGADIR/src/oases	Source files for OASES
$SAGADIR/src/popp	Source files for POPP
$SAGADIR/src/tpem	Source files for TPEM
$SAGADIR/src/prosim	Source files for PROSIM
$SAGADIR/src/prosim	Source files for CPROSIM
$SAGADIR/src/prosim	Source files for ORCA90
$SAGADIR/src/prosim	Source files for GAMA
$SAGADIR/src/ramgeo	Source files for RAMGEO
$SAGADIR/src/obj	Object and library files
$SAGADIR/bin	SAGA scripts and destination of executables
$SAGADIR/examples	Data for inversion with SAGA
$SAGADIR/doc	This document in LaTeX and Postscript format
$SAGADIR/matlab	Some MATLAB files
$SAGADIR/util	Some utility programs

4.3 SAGA platforms

SAGA has been compiled on VMS, Digital UNIX, Linux with ABsoft 3.4 Fortran compiler, Linux with Gnu g77-0.58 Fortran compiler (the compiler does not yet support Fortran structures and thus TPEM was not compiled), Linux with PGI compiler, Linux with the IFC compiler, SunOS, and SGI-Unix.

4.4 Building SAGA

Set your default directory to $SAGADIR/src and edit the **Makefile**. Set the definition of the Fortran compiler and your desired directories for libraries and executables (typically $SAGADIR/bin).

You should decide which forward models you would like to compile. This can easily be selected by commenting them in the "all line" in the **Makefile**. Also the Fortran compiler options should be modified in the **Makefile** in `src` directory. For production runs, it is advisable to compile with a high level of optimization and without array checking.

After performing the changes, compile and link by issuing the command:

```
make all
```

which will generate all SAGA modules.

If the default parameter settings are too large or too small to run a given problem, the parameters may be altered in the parameter include file **comopt.h**. Sometimes the parameters controlling the forward modeling routines should also be changed. See the manuals of the respective forward codes for details.

4.5 SAGA example files

The examples are located in the directory $SAGADIR/examples. Most of the examples are described later in the text. A short description of each example follows, the CPU time is for a DEC Alpha Station 600-5/266.

4.5.1 Forward model OAST

layer11: Described in Sect. 11.1.4. CPU time: 11 s.

tellaro_oast: For comparison with **tellarosnap**. CPU time: 19 min.

simpleoast and **simplesnap**: Simple input files to compare data generated from either OAST or SNAP and inverted using the other model.

4.5.2 Forward model OASR

rfl: Described in Sect. 11.2.2. CPU time: 15 s.

4.5.3 Forward model OASTG

horiz: Described in Sect. 11.3.1. CPU time: 15 s.

4.5.4 Forward model SNAP

sspmisa: Described in Sect. 11.4.4. The example is from the 1993 NRL workshop [46]. CPU time: 8 min.

tellarosnap: Described in Sect. 11.4.5. CPU time: 25 min.

genlmisa_ga: The example is from the 1993 NRL workshop [46]. CPU time: 8 min.

elba: Described in [47], it is similar to the SNAPRD example. CPU time: 4 min.

simplesnap and **simpleoast**: Simple input files to compare data generated from either SNAP or OAST and inverted using the other model.

ys3: Described in Sect. 11.4.6. This example is using matched field inversion on a vertical array based on seven frequencies. The data is from the Yellow Shark experiment. CPU time: 4 h.

4.5.5 Forward model SNAP3D

saga3do: Described in Sect. 12.4.1. **saga3dx**: Described in Sect. 12.4.1. **saga3dx2**: Described in Sect. 12.4.1. **saga3dx3**: Described in Sect. 12.4.1.

4.5.6 Forward model SNAPRD

elbard: Described in Sect. 11.5.3. CPU time: 27 min (three frequencies).

shot05: Described in Sect. 11.5.4. This example is using matched field inversion on a vertical array based on 25 frequencies.

tl_malta_rd: Described in Sect. 11.5.5. The data is range-dependent transmission loss at several frequencies and depths from the Malta experiment. CPU time: 28 h.

4.5.7 Forward model PROSIM

waa: Described in Sect. 11.6.3. The data is from the 1997 Matched Field Inversion Workshop. CPU time: 3 h.

mf: Described in Sect. 11.6.4. This is an example of using matched filter technique.

4.5.8 Forward model POPP

revpopp: Described in Sect. 11.8.2. CPU time: 10 min (one frequency).

rev_grad3: Described in Fig. 22. It uses reverberation data from six frequencies. CPU time: 12 h.

4.5.9 Forward model RAMGEO

tc1horiz: Horizontal array example [74].

tc1v6: Vertical array example [74]. Described in Sect. 11.11.3

4.5.10 Forward model GAMA

map2k: Horizontal array example.

4.5.11 Forward model ORCA90

sdc: Vertical array example using Gibbs sampling and enumerative integration.

4.5.12 Forward model TPEM

tpem_ex: Described in Sect. 11.12.3. CPU time: 10 h.

tpem_rd: An example of a run with range dependent terrain profile.

runs_IEEE96: This directory contains the runs used to generate the figures in Ref. [48]. Note, that because the forward model has changed since the example was generated they cannot be reproduced precisely.

5

SAGA: General features

The inversion package consists of two main programs, an optimization program (SAGA) and a program that analyzes the samples from the optimization (POST). In addition there are several matlab scripts for writing, reading and plotting data.

The saga program can be executed from matlab

>> `runsaga('filename','forward-model')`

This will continuously update the displays for the posteriori analysis while the inversion is running.

The unix command to run SAGA is:

```
saga filename forward-model
```

at present `forward-model` can be either `oast, oasr, oastg, snap, snaprd, popp, prosim, cprosim, orca, gama, ramgeo, tpem`. The filename should be without extension.

The main program stores information which can be processed by the post-processor. The post-processor computes *a posteriori* probability densities, prepares plots of observed data and of data obtained from the inversion. The run command is:

```
post filename forward-model
```

File names are passed to SAGA and POST via environmental parameters. In UNIX systems a typical command file **saga** (in $SAGADIR/bin) is (and a similar one for **post**):

```
#                          the hash mark invokes the C-shell
setenv FOR001 $1.dat   # input file
setenv FOR002 $1.in    # observed data input file
setenv FOR003 $1.wei   # weighting of the data
setenv FOR007 $1.out   # output file
setenv FOR007 $1.pout  # output file from post
setenv FOR009 $1.eof   # shape function file
```

```
setenv FOR010 $1.mat   # The best obtained parameters and
                       #  objective function for each population.
setenv FOR011 $1b.mat  # The $1.mat sorted according to fit (POST)
setenv FOR011 $1.m     # plotting parameters used by plotsaga (POST)
setenv FOR019 $1.plp   # plot parameter file
setenv FOR020 $1.plt   # plot data file
setenv FOR028 $1.cdr   # contour plot parameter file
setenv FOR029 $1.bdr   # contour plot data file
setenv FOR030 $1.obs   # synthetic generated from the initial environment
setenv FOR060 $1.ext   # same format as $1.mat, for each improved fit

 if     ($argv[2] == "snap"    ) then
      $SAGADIR/bin/sagasnap                 # snap   executable
 elseif ($argv[2] == "snaprd"  ) then
      $SAGADIR/bin/sagasnaprd               # snaprd executable
 elseif ($argv[2] == "oast"    ) then
      $SAGADIR/bin/sagaoast                 # oast   executable
 elseif ($argv[2] == "oasr"    ) then
      $SAGADIR/bin/sagaoasr                 # oasr   executable
 elseif ($argv[2] == "oastg"   ) then
      $SAGADIR/bin/sagaoastg                # oast gradient executable
 elseif ($argv[2] == "popp"    ) then
      $SAGADIR/bin/sagapopp                 # popp   executable
 elseif ($argv[2] == "tpem"    ) then
      $SAGADIR/bin/sagatpem                 # tpem   executable
 elseif ($argv[2] == "prosim"  ) then
      $SAGADIR/bin/sagaprosim               # prosim executable
 elseif ($argv[2] == "cprosim" ) then
      $SAGADIR/bin/sagacprosim              # cprosim executable
 elseif ($argv[2] == "ramgeo"  ) then
      $SAGADIR/bin/sagaramgeo               # ramgeo executable
 elseif ($argv[2] == "orca"  ) then
      $SAGADIR/bin/sagaorca               # ramgeo executable
 elseif ($argv[2] == "gama"  ) then
      $SAGADIR/bin/sagagama               # ramgeo executable
 endif
```

Each time POST is executed it updates the files **results** and **results.m** in the current directory.

5.1 Graphics

Standard output files from POST can be plotted using MATLAB. Simply put **$SAGA/MATLAB** in your MATLAB path, for example in the .login file:

```
setenv MATLABPATH $home/saga/MATLAB
```

start up MATLAB and execute

```
>> plotsaga
```

When plotting from MATLAB, each *a posteriori* distribution is plotted individually and when option **p** is specified a plot of the observed data and data corresponding to the best fit is also provided.

A scatter plot of the objective function for each parameter can the obtained by executing.

```
>> sagascat
```

Contour plots (of ambiguity function, 2D marginal PPD, and Cramer-Rao bounds) can also be plotted using MATLAB:

```
>> contsaga
```

`contsaga` uses the data values in the *.bdr file and some commands in the *.m file; the *.cdr file is neglected when plotting from MATLAB.

The files *.m and *.mat from the program can be plotted using MATLAB.

5.2 Useful scripts

Some useful UNIX-scripts are provided in the $SAGADIR/bin:

`zap name`
Kills all processes that has `name` in their command-field, e.g. `zap saga`.

`zapnice name`
For all processes with `name` in their command-field the CPU-priority is reduced by one.

`zapmax name`
For all processes with `name` in their command-field the CPU-priority is reduced to the minimum.

6
SAGA: The input file

The structure of the input file is given in Table 1, the file must have the extension *.dat.

6.1 SAGA options

The following options are supported by SAGA. Each option should be separated by a blank space. The pointer (`iopt`) is used internally in the program; it is of use only to the programmer.

FORWARD MODEL
(the forward model is specified at execution time, the following flags are set automatically):

(iopt(30)=1) SNAP is used.
(iopt(30)=2) SNAPRD is used.
(iopt(30)=3) OAST is used.
(iopt(30)=4) OASR is used.
(iopt(30)=5) POPP is used.
(iopt(30)=6) TPEM is used.
(iopt(30)=7) PROSIM is used.
(iopt(30)=8) CPROSIM is used.
(iopt(30)=9) RAMGEO is used.
(iopt(30)=10) GAMA is used.
(iopt(30)=11) ORCA is used.

(iopt(12)=1) 3 indexes are used for pointing to the optimization parameters (used in SNAPRD, TPEM, PROSIM, CPROSIM, GAMA,ORCA, RAMGEO).
(iopt(12)=0) 2 indexes are used for pointing to the optimization parameters (used in SNAP, OASR, OAST, POPP).

INVERSION DOMAIN
(default **r**)

w (iopt(1)=1) (for OAST) inversion in wavenumber–depth domain.
r (iopt(1)=2) (for OAST) inversion in range–depth domain.

Input parameter	Description
Block I: TITLE (1 line)	
TITLE	Title of run
Block II: OPTIONS (1 line)	
A B C ···	Output and computational options, see Sect. 6.1
Block III: GA- PARAMETERS (2 lines)	
niter q npop ...	niter: Forward modeling runs for one population q: Population size npop: Number of parallel populations ...: Some options require more parameters (`t0`,...)
px pu pm	px: Crossover rate (typically 0.8) pu: Update rate (typically 0.5) pm: Mutation rate (typically 0.05)
Block IV: FORWARD MODEL PARAMETER (many lines)	
. . . .	This block correspond to the forward model in the same format as given in the users manual for each forward model. See Sect. 11
Block V: OPTIMIZATION PARAMETER (nparm+1 lines)	
nparm parm index Fmin Fmax Ndiscr	nparm: Number of parameters to optimize parm: Pointer to physical parameter, see Sect. 11 index: Addresses a specific variable within a vector (For SNAPRD, PROSIM and TPEM: `index` represents two variables `index`, `index2`, see Sect. 11) Fmin: Lower bound for the search parameter Fmax: Upper bound for the search parameter Ndiscr: Number of discrete values of the parameter

Table 1 SAGA *input file structure.*

(iopt(1)=3) (for OASR, automatic) inversion in angle domain.
(iopt(1)=4) (for POPP, automatic) inversion in time domain.

OBSERVED DATA FORMAT
(See also Sect. 7)
(Should **always** *be specified)*:

- **d** (iopt(2)=1) reading data (subroutine `readdat2`). This is the most general format, which allows for reading complex-valued or real-valued data as a function of frequency, depth and range. For the format of the *.in file see Sect. 7.2.
- **D** (iopt(2)=2) reading data (subroutine `readdata`). This is only for data given as a function of range. The range block in the forward model should then contain `Rmin` and `Rmax` in one line. For the format of the *.in file see Sect. 7.3.

- **e** (iopt(2)=3) reading complex pressure vector on a *vertical* array (subroutine `read_HP`). This option also requires option **f** or option **F** for objective function. For the format of the *.in file see Sect. 7.4.
- **T** (iopt(2)=3) reading complex pressure vector on a *horizontal* array (subroutine `read_HP`). This option also requires option **k** for objective function. Note, that the `snap3d` is rotating a vertical array and therefore option **e** should always be used. For the format of the *.in file see Sect. 7.5.
- **c** (iopt(13)=1) reading covariance matrix (subroutine `read_cov`). For the format of the *.in file see Sect. 7.1.
- **W** (iopt(21)=1) writing calculated data using the baseline model to the *.obs file. The format is determined by the input option **d, D, c, e**. The *.obs file can then be copied to the *.in file and used as 'observed' data. Even when the input file does not exist and the program aborts, the *.obs file is written, and the *.obs file can then be copied to the *.in file.
- **z** (iopt(11)=1) Adds noise to the covariance matrix (option **c**) before it is written to the *.obs file. It is only added in the diagonal and with a specific SNR. The SNR is specified in dB and is given as last parameter in the second line with GA-parameters, after the mutation rate `pm`.

SCALING OF COVARIANCE MATRIX
(default is no scaling)

- **b** (iopt(20)=1) the covariance matrix is normalized by dividing it with the sum of the diagonal. The maximum obtainable Bartlett power is slightly less than one (depending on the noise in the data).
- **B** (iopt(20)=2) the covariance matrix is normalized by the largest eigenvalue of the covariance matrix. Thereby, the maximum obtainable Bartlett power is one.

TRANSFORMATION OF CALCULATED DATA
(default is no transformation)

- **s** (iopt(3)=1, itrans(1)=1) weighting replica with $\sqrt{r}$. The observed data should be weighted before running SAGA. This option should only be used for transmission loss data. In this way the transmission loss is corrected for geometrical spreading.
- **R** (iopt(3)=2, itrans(2)=3) weighting replica with $1/\sqrt{r}$. The observed data should be weighted before running the program. Should only be used for transmission loss data. This places more emphasis on matching the closer range of the transmission losses.
- **l** (iopt(3)=1, itrans(3)=2) inversion with dB scaled observed and calculated data. The observed data should be expressed as $-20 \log p$.
- **l2** (iopt(3)=12, itrans(3)=12) inversion with dB scaled observed and calculated data. The observed data should be expressed in physical units, not in dB.
- **G** (iopt(3)=5, iopt(25)=1, itrans(5)=5) using only the magnitude of observed and calculated data.
- **h** (iopt(27)=1) observed and calculated data are normalized to unity. This works only for a local method; for the global method it is defined through the objective function.
- **M** (iopt(3)=44, itrans(4)=44) both data and replica are weighted with the values read from the *.wei data file, see Sect. 7.6.
- **M1** (iopt(3)=40, itrans(4)=40) only data are weighted with the values read from the *.wei data file, see Sect. 7.6.
- **M2** (iopt(3)=04, itrans(4)=04) only replica are weighted with the values read from the *.wei data file, see Sect. 7.6.

OBJECTIVE FUNCTION
(See also Sect. 8)
(Should **always** *be specified)*

Objective function that works with data formats option **D, d, e, T**

- **N** (iopt(5)=0), Eq. (19), objective function is computed based on the sum of the squared error between the magnitude of the observed and calculated data using complex-valued data (if available). This matches only the shape of one curve using all frequency, range and depth information.
- **n** (iopt(5)=1), Eqs. (24, 25), objective function is computed based on the sum of the squared error between the magnitude of the observed and calculated data as a function of range. This matches only the shape of a curve. There are as many curves as frequencies and depths.
- **X** (iopt(5)=3), Eqs. (24, 26), objective function is computed based on the sum of the squared error between the magnitude of the observed and calculated data. This matches both the shape and the offset of a curve. There are as many curves as frequencies and depths.

Objective function that works with data format option **e, d, T** (complex-valued data). The standard is to use option **e** for vertical array. Except for SNAP where horizontal arrays are obtained by rotating a vertical array, horizontal array data is read in using the underlineoption **T**

- **f** (iopt(5)=4), Eq. (20), incoherent addition of Bartlett power across frequencies using pressure vector on an array(s). This array is summed over the depth of the array, but for SNAP3D the array can be arbitrary oriented. At least two phones required. Each frequency component has is weighted according to the power in the received signal.
- **f1** (iopt(5)=4, isubopt(5)=1), Eq. (21). Similar to option **f** but in the summation of the objective function each frequency component has the same weight. Both replica and data vector are normalized to one.
- **k** (iopt(5)=7), Eq. (20), incoherent addition of Bartlett power across frequencies using pressure vector on an horizontal array(s). This array is summed over the range of the array, At least two phones required. Each frequency component is weighted according to the power in the received signal.
- **k1** (iopt(5)=7), Eq. (20). Similar to option **k**, but in the summation of the objective function each frequency component has the same weight.
- **F** (iopt(5)=5), Eq. (22), broadband matched filter. Incoherent addition of Bartlett power across phones using pressure vector for several frequencies. At least two frequencies required. This option works best in combination with PROSIM or CPROSIM. When combining several phones each phones has the same weighting, regardless of the received energy on the phone.
- **j** (iopt(5)=6), Eq. (23), this objective function compares the relative phase and magnitude of the observed and calculated pressure vector. Used in Ref [73].

Objective function that works with data format option **c** (covariance matrix).

- **c** (iopt(13)=1), Eq. (27), the objective function is the Bartlett power. By default the Bartlett power for each frequency is summed.
- **O** (iopt(18)=1), Eq. (28), the Bartlett power for each frequency is multiplied together.

Objective function is modified, this is independent of data format.

- **L** (iopt(9)=1) regularization, see Annex B. This option has influence on the computation of the objective function in SAGA. The objective function then consists of two parts, the standard one measuring the fit of the data and one measuring the deviation from the baseline model. For this option we must first specify for which parameters the *a priori* information should be included. In Block IV, Table 1, after `Ndiscr` a positive number indicates *a priori* information and a negative indicates none.

q (iopt(23)=1) *a priori* information, see Annex B. This option does not affect the computation of the objective function in SAGA, but only the computation in POST. For this option we must first specify for which parameters the *a priori* information should be included. In Block V, Table 1, after `Ndiscr` a positive number indicates *a priori* information and a negative indicates none. The *a priori* information used for each parameter is a triangular distribution with maximum at the baseline value and zero at the bounds.

V (iopt(31)=1) In calculating the objective function the code is not stopped in case of an unphysical response function, where all data is zero. This should only be used to circumvent problems in the forward code.

OPTIMIZATION
(default is to use genetic algorithms)

E (iopt(17)=1) shape functions is used to describe the environmental input. A shape function file must be supplied, see Sect. 9.

x (iopt(24)=1) the baseline model in the input is included as an initial member of the first `t0` populations in the GA optimization.

a (iopt(4)=1) using simulated annealing for the optimization. The version is adapted from a code by Mike Collins. `niter` is the number of Metropolis steps, the number of forward models is then `niter*nparm`. Two additional parameters should be specified in the first line of the GA parameter block: `t0` is starting temperature (often 1) and `t1` indicating that quenching is used after iteration `t1`. `Ndiscr` determines the maximum allowed jump (often `Ndiscr = 1`). There is no post-processing when using simulated annealing, only the best model is displayed.

v (iopt(4)=3) using very fast simulated annealing (VFSA) for the optimization. The version is adapted from a code by M. Sen and based on the paper by Ingber [21]. It seems to be faster than the simulated annealing version described above. VFSA offers great flexibility in selecting optimization control parameters, but here only the initial temperature can be varied. `niter` is the number of Metropolis steps, the number of forward models is then `niter*nparm`. One additional parameters should be specified in the first line of the GA-parameter block: `t0` is starting temperature (often 1). There is no post-processing when using simulated annealing, only the best model is displayed.

g (iopt(4)=2) using the Powell method for the optimization. The Powell method is very useful as it does not require any computations of gradients. Gauss–Newton can also be used, but at present only available using the analytic derivatives OASTG. The starting point for the local search is the baseline model given in the input. The stop criteria is that the improvement for each parameter is less than $0.2(F\max - F\min)/N\text{discr}$.

H (iopt(16)=1) the hybrid optimization scheme is used. At present only available using the analytic derivatives OASTG. A forward modeling call consists then of 5 Gauss–Newton steps, see [18].

Z (iopt(29)=1) Single point crossover. In SAGA the default is to do multiple point crossover (one crossover for each parameter), thus the value of all parameter may be changed in one forward model call. The single point crossover only changes one parameter per forward model call.

y (iopt(33)=1) Gray coding [44] of the binary numbers. Gray code represents each number in the sequence of integers $[0, ..., 2^{N-1}]$as a binary string of length N in an order such that adjacent integers have Gray code representations that differ only one bit positions. Marching through the integer sequence therefore requires flipping just one bit at a time. Some researches have found that Gray code representation works better than binary representation.

? (iopt(34)=1) A true random number is used instead of using the same random number when starting the random number generator.

LIKELIHOOD INTEGRATION SAGA

The default is to estimate the posterior probabilities based on the models sampled during a GA run (this is done from POST. That approach is very efficient and gives a good indication of the posteriori probabilities. It is quite empirical, biased and not very precise. For details on the likelihood integration see Chapter **XXX**.

The likelihood integration module is invoked by using Option **S** and on a new line below the GA parameters the user must specify

```
nu e_stop e_rot k_grow
```

`nu` is the error in Eq. (**??**).
`e_stop` is the stop criteria (recommended value 0.1)
`e_rot` is the criteria for accuracy of the matrix before rotating (recommended value 0.1)
`k_grow` is the factor used in Eq. (**??**).

The output from the gibbs sampling is written to the binary files *filenm.gs1* and *filenm.gs2*. These can be read and plotted with *plotgibbs.m* (it also uses *read_gs_bin.m*). the enumerative integration is written to the binary file *enum.mat* and can be read and plotted with *plotenum.m* (it also uses *read_enum_bin.m*).

S or **S0** (iopt(4)=1, isubopt(4)=1) Gibbs sampling. During optimization all parameters are changed simultaneously. `niter` iterations is carried out and convergence is then tested. If the

S1 (iopt(4)=1, isubopt(4)=0) Gibbs sampling. During optimization one parameters at a time is changed. Otherwise the same as option **S**.

Sx1 (isubopt(35)=1) Using an adaptive adjustment of the search interval, based on Eq. (**??**).

S2 (iopt(4)=1, isubopt(4)=2) Enumerative integration.

PLOTS FROM POST
(default is to plot the *a posteriori* probability distribution on a scale corresponding to the search interval using the default weighting. A good data check is obtained by plotting the data as well as the best model using option **p**. A table of best fit, mean and most likely value for each parameter is given in the output.)

p (iopt(10)=1) a line plot of the data (full line) and the calculated data (dashed line) using the best found model. For a covariance matrix as input data, option **c**, we estimate the pressure vector as the first eigenvector of the covariance matrix. For a vertical array the power across the array is only plotted, but the phase and Bartlett power can also be plotted.

p2 (iopt(10)=2) For the covariance matrix option option **c** both magnitude and phase is plotted (two plots per frequency).

p3 (iopt(10)=3) For covariance matrix option option **c** (iopt(10=3) For covariance matrix (opt c) both magnitude and phase and 'Bartlett power' is plotted (three plots per frequency). The precise definition of 'Bartlett power' is in [59]

A (iopt(19)=1) the starting model is superimposed on the *a posteriori* probability plot.

i (iopt(14)=1) plotting the *a posteriori* probability distributions in physical units, instead of plotting them all on a dimensionless scale from 0 to 1.

I (iopt(14)=0) plotting the *a posteriori* probability distributions on a dimensionless scale from 0 to 1. This creates one plot of the distributions.

m13 (iopt(16=1) computes the 2-D marginal *a posteriori* density distribution between two parameters, here parameters 1 and 3. Both parameters have to be one digit, i.e. less than 10. The plot is normalized so that the maximum is one. Usually a nice plot is obtained by using the scale 0–0.1. It can be plotted using CPLOT.

u (iopt(28)=2) the *a posteriori* probability distribution is produced using a uniform weighting of all the observed model vectors. This causes a flatter distribution.

Pj (iopt(28)=1) maximum likelihood based post-processing of the *a posteriori* probability distribution. It is based on Monte Carlo integration of the likelihood function. j indicates the number of modes used in the likelihood function; if it is not specified then $j = 5$. This requires the use of option **c** or **f** as choice of objective function.

PLOTS FROM SAGA:
For these options no global optimization is carried out and it is not necessary to run POST.

C (iopt(8)=1) contour plot of the objective function versus first and second parameter given in the inversion. The first search parameter specifies the x-axis and the second the y-axis. The discretization along each axis is specified by `Ndiscr`. The maximum discretization in each direction is 200. It can be plotted using the matlab program `contsaga`. The objective function is expressed in dB relative to the minimum of the in the objective function 10log10(Φ/Φ_0), which will then have a maximum value of 0 dB. The dynamic scale will be 5 dB.

C1 (iopt(8)=2) Same as option **C**, except that the objective functions is not scaled relative to the maximum of ambiguity function. $-10\log10(\Phi)$ is plotted.

C2 (iopt(8)=3) Same as option **C**, except one minus the objective functions is plotted and the plot is not scaled. $-10\log10(1-\Phi)$ is plotted. This is the recommended approach when using a conventional processor.

C3 (iopt(8)=4) Same as option **C**, except one minus the objective functions is plotted and the plot is not scaled. $-10\log10(1/N * \sum_i(trC_i) - \Phi)$ is plotted.

U (iopt(22)=1) Uncertainty for a **local method**. Based on analytic computed gradients. The Cramer-Rao bounds for a stochastic signal is computed. As a function of the first two unknown parameters the standard deviation of the diagonal entries in the Cramer-Rao matrix is plotted for all unknown parameters. It requires that `oastg` is used, with option option **g**.

K (iopt(32)=1) Line plot of objective function. Using the reference environment as a baseline a line plot of the sensitivity for each optimization parameter specified in the input file is computed using the bounds and discretization as given in the input file. The plot is given on a dB-scale with the maximum for each parameter being normalized to 0 dB. The plot shows the variation of $-10log10(\phi)$, where ϕ is the objective function.The plot is obtained using MATLAB on the *filename.m*-file.

K1 (iopt(32)=2) Similar to option **K**, but the maximum for each parameter is not normalized. $-10\log10(\Phi)$ is plotted.

K2 (iopt(32)=3) Same as option **K**, except one minus the objective functions is plotted and the plot is not scaled. $10\log10(1-\Phi)$ is plotted.

MISCELLANEOUS:

Q (iopt(6)=1) debugging. This option writes lots of extra information for the first couple of iterations.

7

Format of the observed data

The data file (*.in) contains information about the observed data. Even for synthetic data, this is the standard mode to pass observations to the SAGA program. Data can be written out from the program by using option **W**. The program supports four formats (**c**, **d**, **D** and **e**) which are described below. For other inversion purposes they might have to be modified.

For each of these files a "**!**" as the first character of a line means a comment. No blank lines!

The order of the data in the *.in file should always be the same as the specified in the *.dat file. For example often the receiver depths in the *.dat can be specified from the top or the bottom of the array, it should then be ordered similarly in the *.in file.

7.1 Covariance matrix file

The covariance matrix file stores the covariance matrix calculated from the pressure on a vertical array for a single frequency. The program gives a warning if the first line does not contain the string "Covariance matrix". The format of the file is:

```
loop over freq                    ! in increasing order
     loop over range            ! ranges as appearing in input file
      read(2,*)dummy-title
      read(2,*)dummy-freq
      read(2,*)dummy-ndep
      do idep=1,ndep
         READ(2,*)receiver_depth                 !rd
      enddo
      do idep1=1,ndep                           ! Row
         do idep=1,ndep                         ! Column
             READ(3,*)jdum,idum,cov(idep1,idep,ibart)
                                                ! idum, jdum is not used
         enddo
      enddo
   enddo
```

```
enddo
```

In this file, the covariance matrix for several frequencies can be stored, not all covariance matrices need to be read in. The program only stores the ones corresponding to the frequencies to be used in the inversion. The covariance matrices should be given in the same order as the frequencies in the input file.

If the first or last receiver depth does not correspond to the depth in the input file a warning is issued.

7.2 General data input (option d)

This is a general data file for passing data to the SAGA program (using subroutine `readdat2`). The format is (data can be either real-valued or complex-valued):

```
loop over freq                    ! in increasing order
   loop over depth
      loop over ranges
         read(2,*)ifreq idepth,irange,data
      enddo
   enddo
enddo
```

where `ranges` can mean either number of ranges or wavenumbers. The counters *ifreq idepth, irange* is not used when reading the data. There is no check if the correct frequencies, depths or ranges are read.

7.3 Range data input (option D)

This is for reading in pressure-data as a function of range (using subroutine `readdata`). Only points in the range `Rmin` to `Rmax` are used. `Rmin` and `Rmax` are given in the input file, units is meters. The format is (data must be real-valued):

```
reads to End_Of_File:
do i=1,10000
    read(2,*) rng,  (press(i,j),j=1,ncurv)
enddo
```

where `[1...ncurv]` consists of an outer loop over frequency and an inner loop over depth.

7.4 Vertical array data (option e)

The vertical array file stores the pressure on a vertical array for several frequencies (using subroutine `readHP`). The format of the file is similar to the covariance matrix format except for the inner do-loop. The program gives a warning if the first line does not contain the string "Hydrophone vectors". The format is:

```
loop over range         ! as appearing in input file
   loop over freq                    ! in increasing order
      read(2,*)dummy-title
      read(2,*)dummy-freq
      read(2,*)dummy-ndep
      do idep=1,ndep
         READ(2,*)receiver_depth                 !rd
      enddo
      do idep=1,ndep
         READ(2,*)idum,pres(idep,ibart) ! complex-valued pressure
      enddo
   enddo
enddo
```

In this file, the pressure vector for several frequencies can be stored, not all vectors need to be read in. The program only stores the ones corresponding to the frequencies to be used in the inversion.

There is no check when reading the ranges.

If the first or last receiver depth does not correspond to the depth in the input file a warning is issued.

7.5 Horizontal array data (option T)

The vertical array file stores the pressure on a vertical array for several frequencies (using subroutine `readHA`). The format of the file is similar to the covariance matrix format except for the inner do-loop. The format is:

```
loop over depth         ! as appearing in input file
   loop over freq                    ! in increasing order
      read(2,*)dummy-title
      read(2,*)dummy-freq
      read(2,*)dummy-ndep
      do iran=1,nrange
```

```
         READ(2,*)receiver_depth                  !rd
      enddo
      do iran=1,nrange
         READ(2,*)idum,pres(iran,ibart) ! complex-valued pressure
      enddo
   enddo
enddo
```

In this file, the pressure vector for several frequencies can be stored, not all vectors need to be read in. The program only stores the ones corresponding to the frequencies to be used in the inversion.

There is no check when reading the ranges.

If the first or last receiver depth does not correspond to the depth in the input file a warning is issued.

7.6 Weighting file (option M)

The filename is *.wei. The purpose of this is to weight the data according to *a priori* knowledge by a user-specified function, instead of those supplied by the program $(1/\sqrt{r},\ \sqrt{r},\ -20\log p)$. The use of a simple function is recommended. The format is similar to option **e**

```
loop over range         ! as appearing in input file
   loop over freq                    ! in increasing order
      read(2,*)dummy-title
      read(2,*)dummy-freq
      read(2,*)dummy-ndep
      do idep=1,ndep
         READ(2,*)receiver_depth                  !rd
      enddo
      do idep=1,ndep
         READ(2,*)idum,weight(idep,ibart) ! complex-valued weight
      enddo
   enddo
enddo
```

8

The objective function

It is important to use the correct Fourier transform pair when transforming observed data from time to frequency domain. If the wrong sign convention is used the inversion will be meaningless. We are using the following transform pair:

$$p(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} P(t)\mathrm{e}^{-i\omega t}\,\mathrm{d}t \ , \tag{13}$$

$$P(t) = \int_{-\infty}^{\infty} p(\omega)\mathrm{e}^{i\omega t}\,\mathrm{d}\omega \ . \tag{14}$$

This corresponds to matlab's $p(\omega) = \ \mathrm{FFT}(P(t))$.

One of the most important tasks in carrying out an inversion by means of optimization is to define a proper objective function. Here we have selected some of the most common.

Let p and q be the observed and calculated data, respectively. They are complex matrices and contain a number of frequencies (N_{freq}), depths (N_{dep}) and ranges or wavenumbers (N_{x}).

8.1 Transformation of data

Before the objective function is calculated it is possible to transform the calculated data to another form that might be more suitable for a particular inversion. The objective function will then be applied directly on the transformed data.

For option **G** the magnitude of both the computed and observed data is taken

$$p_{ijk} = |p_{ijk}|, \quad q_{ijk} = |q_{ijk}| \tag{15}$$

$$\text{for} \qquad i = 1,\ldots,N_{\mathrm{freq}}\ , \quad j = 1,\ldots,N_{\mathrm{dep}}\ , \quad j = 1,\ldots,N_{\mathrm{x}} \tag{16}$$

For option **l** only the computed data are transformed to a dB scale. The observed data are assumed to be already on a dB scale.

$$p_{ijk} = -20\log|p_{ijk}|, \quad q_{ijk} = -20\log|q_{ijk}| \tag{17}$$

$$\text{for} \qquad i = 1,\ldots,N_{\mathrm{freq}}\ , \quad j = 1,\ldots,N_{\mathrm{dep}}\ , \quad j = 1,\ldots,N_{\mathrm{x}} \tag{18}$$

8.2 Direct observations

The most general objective function is option N. $\hat{p}$ and $\hat{q}$ are the data normalized to unity[1]:

$$\phi = |1 - \sum_{i=1}^{N_{\text{freq}}} \sum_{j=1}^{N_{\text{dep}}} \sum_{k=1}^{N_{\text{x}}} \hat{p}_{ijk}^{*} \hat{q}_{ijk}| \ . \tag{19}$$

This corresponds to the mean squared error.

Option **f** is the incoherent sum of the Bartlett power for each frequency. Both observed and computed data are based on vectors and not the usual correlation matrix:

$$\phi_{\text{f}} = \frac{1}{N_{\text{x}} N_{\text{freq}}} \sum_{k=1}^{N_{\text{x}}} \sum_{i=1}^{N_{\text{freq}}} \left[\sum_{j=1}^{N_{\text{dep}}} |p_{ijk}|^2 - \frac{|\sum_{j=1}^{N_{\text{dep}}} p_{ijk}^{*} q_{ijk}|^2}{\sum_{j=1}^{N_{\text{dep}}} |q_{ijk}|^2} \right] \ . \tag{20}$$

This objective function weights each frequency component according to the received signal. Under simplifying assumptions this is the maximum likelihood objective function, see Appendix A.

Option **f1** is the incoherent sum of the Bartlett power for each frequency. But each frequency component has the same weight in the objective function. This works best if it is known that the source signal is nearly flat. Both observed and computed data are based on vectors and not the usual correlation matrix:

$$\phi_{\text{f1}} = 1 - \frac{1}{N_{\text{x}} N_{\text{freq}}} \sum_{k=1}^{N_{\text{x}}} \sum_{i=1}^{N_{\text{freq}}} \frac{|\sum_{j=1}^{N_{\text{dep}}} p_{ijk}^{*} q_{ijk}|^2}{\sum_{j=1}^{N_{\text{dep}}} |p_{ijk}|^2 \sum_{j=1}^{N_{\text{dep}}} |q_{ijk}|^2} \ . \tag{21}$$

This objective function weights each frequency component identical.

Option **F** is a matched filter. We sum incoherently over depth and range, but coherently over frequency.

$$\phi_{\text{F}} = 1 - \frac{1}{N_{\text{x}} N_{\text{freq}}} \sum_{k=1}^{Nx} \sum_{j=1}^{N_{\text{dep}}} \frac{|\sum_{i=1}^{N_{\text{freq}}} p_{ijk}^{\text{T}} q_{ijk}|^2}{\sum_{i=1}^{N_{\text{freq}}} |p_{ijk}|^2 \sum_{i=1}^{N_{\text{freq}}} |q_{ijk}|^2} \ . \tag{22}$$

Classical matched filter is done in the time domain by correlating the observed time series and the synthetic time series. The above expression in the frequency domain is similar to classical matched filtering.

Option **j**

Here, we assume the magnitude $|S_l|$ of the source signal is known but the phase of the source signal is unknown. The closed form solution for the source strength

[1] * is the conjugate operator

then is Both observed and computed data are based on vectors and not the usual correlation matrix:

$$\phi_{\mathrm{j}} = \frac{1}{N_{\mathrm{x}} N_{\mathrm{freq}}} \sum_{k=1}^{N_{\mathrm{x}}} \sum_{i=1}^{N_{\mathrm{freq}}} \left[\sum_{j=1}^{N_{\mathrm{dep}}} |p_{ijk}|^2 + |S_i|^2 \sum_{j=1}^{N_{\mathrm{dep}}} |q_{ijk}|^2 - 2|S_i| \left| \sum_{j=1}^{N_{\mathrm{dep}}} p_{ijk}^* q_{ijk} \right| \right] . \tag{23}$$

This objective function weights each frequency component according to the received signal.

For option **n** and **X** we use only the magnitude of the observations:

$$\phi_{\mathrm{nX}} = \sum_{i=1}^{N_{\mathrm{freq}}} \sum_{j=1}^{N_{\mathrm{dep}}} \sum_{k=1}^{N_{\mathrm{x}}} (|p_{ijk}| - A_{ij}|q_{ijk}|)^2 , \tag{24}$$

where for option **n** we correct for the offset of the curve

$$A_{\mathrm{n}ij}^2 = \frac{\sum_{k=1}^{N_{\mathrm{x}}} |p_{ijk}|^2}{\sum_{k=1}^{N_{\mathrm{x}}} |q_{ijk}|^2} \tag{25}$$

and for option **X** there is no offset correction,

$$A_{\mathrm{X}} = 1 . \tag{26}$$

This should be used if the data is well calibrated, so that the absolute levels of the Transmission loss is known.

8.3 Covariance matrix

For option **c** we use the Bartlett power. For a vector of pressure observations at frequency i, at range k, on a vertical array p_{ij} $(j = 1, N\mathrm{dep})$, the covariance matrix for a given frequency and range is constructed $R_{jl,ik} = p_{ijk} p_{ilk}{}^{\mathrm{T}}$. For observed data, the covariance matrix is based on the ensample average of the estimated covariance matrix formed for each time snapshot. Some help in the construction of this covariance matrix is available in the MATLAB directory. The MATLAB routines also writes the covariance matrix out in SAGA format. This matrix is read from the *.in file. q_{ij} $(j = 1, N\mathrm{dep})$ is a vector of calculated pressure on the vertical array,

Figure 6 illustrates the features of the cross-correlation matrix at 400 Hz with the receiving array at a range of about 1 km away from the source for an experiment in the Strait of Singapore [17]. At this frequency, the signal to noise ratio is approximately 14 dB. At a bin width of 1.45 Hz, most of the energy is contained within that band and the adjacent bins have much smaller levels.

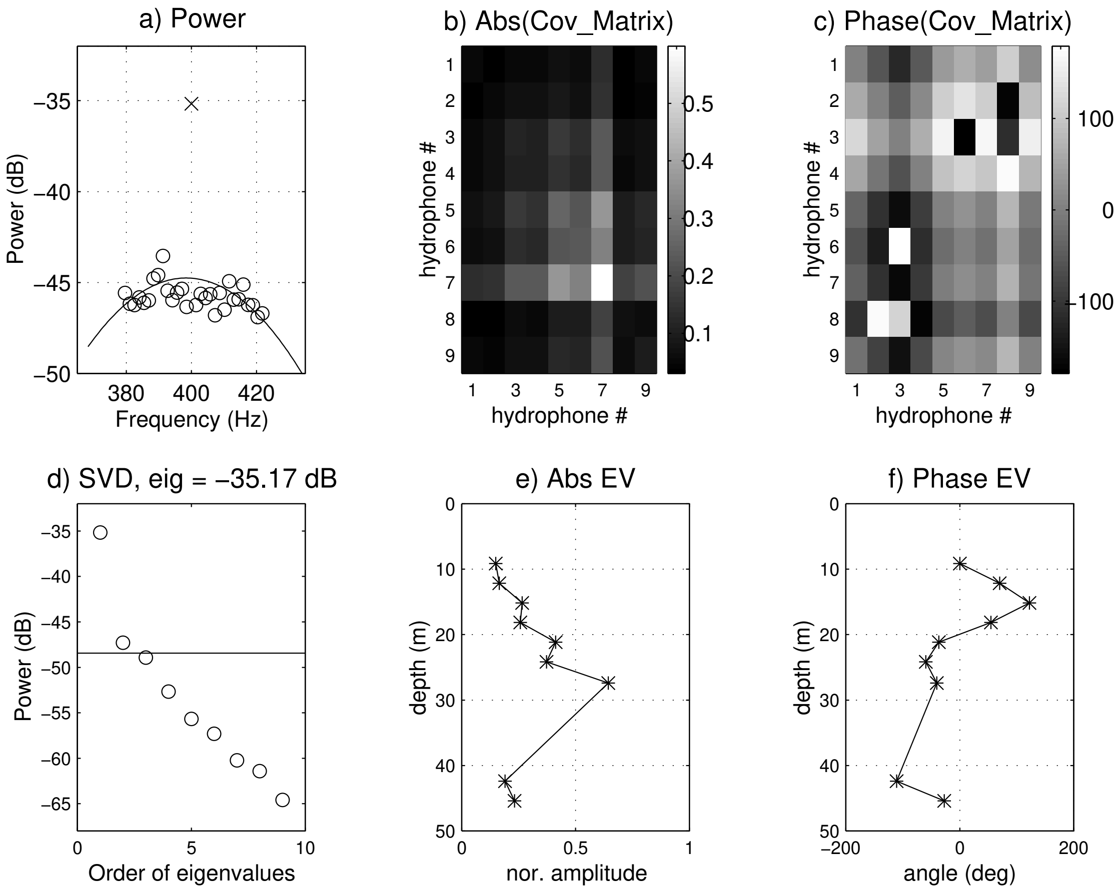

Figure 6 *Analysis of a received signal. This is used for selecting the frequencies and for quality control of the data. First the cross-correlation matrix is formed for several bins in a selected frequency interval. Based on the first eigenvalue of the cross-correlation matrix at each bin (a) the frequency with the highest power is selected [marked with a cross], the corresponding magnitude (b) and phase (c) of the this correlation matrix is plotted. All the eigenvalues for this cross-correlation matrix (d) is then plotted; the estimated noise floor (solid line) is from a source-free section of the time series. The SNR is seen to be about 14 dB. Based on the cross-correlation matrix, magnitude (e) and phase (f) of the first eigenvector is plotted. Figure 6e can be used for detecting unusual gain for a hydrophone and Fig. 6f can be used to detect a* $180°$ *phase shift of the signal on a receiver.*

$$\phi_{\mathrm{c}} = \sum_{i=1}^{N_{\mathrm{freq}}} \sum_{k=1}^{N_{\mathrm{x}}} \sum_{j=1}^{N_{\mathrm{dep}}} \left[R_{jj,ik} - q^*_{ijk} \sum_{l=1}^{N_{\mathrm{dep}}} R_{jl,ik} q_{ilk} \right] \quad . \tag{27}$$

It can be shown that the above formula is a maximum likelihood estimate if the noise is assumed Gaussian with known or constant mean, see e.g. [45]. If the noise is unknown and frequency dependent then option **O** provides a maximum likelihood estimate; it is obtained by replacing the summation in the above formula with a product,

$$\phi_{\mathrm{c}} = \prod_{i=1}^{N_{\mathrm{freq}}} \prod_{k=1}^{N_{\mathrm{x}}} \sum_{j=1}^{N_{\mathrm{dep}}} \left[R_{jj,ik} - q^*_{ijk} \sum_{l=1}^{N_{\mathrm{dep}}} R_{jl,ik} q_{ilk} \right] \quad . \tag{28}$$

This corresponds to summing the logarithmic powers (dB). From a practical point of view no significant difference has yet been found between the two objective functions, Eqs. (27) and (28).

8.4 Cramer-Rao bound

In order to assess the performance of the different estimation procedures to find a set of parameter $\mathbf{m}$ the covariance matrix of the estimation errors as expressed by the Cramer Rao lower Bound (CRLB) is used, see eg. Ref. [51, 50]..

For an unbiased estimate $\hat{\mathbf{m}}$ of a real parameter vector $\mathbf{m}_0$, based on observations $\mathbf{p}$. The Cramer-Rao lower bound on the estimation error covariance is given by

$$\mathbf{C}^{\mathrm{CR}} = \mathrm{E}\left[(\hat{\mathbf{m}} - \mathbf{m}_0)(\hat{\mathbf{m}} - \mathbf{m}_0)^{\mathrm{T}} \right] = \mathbf{F}^{-1} \tag{29}$$

where the Fisher information matrix $\mathbf{F}$ is given by

$$\mathbf{F} = \mathrm{E}\left[\frac{\partial^2 \log \mathcal{L}(\mathbf{p}|\mathbf{m})}{\partial \mathbf{m} \partial \mathbf{m}} \right]$$

where $\mathcal{L}$ is the likelihood Function.

The CRLB provides a performance ceiling beyond which an optimal estimation procedure cannot exceed, thus for an estimated set of parameters $\hat{\mathbf{m}}$

$$\mathrm{E}\left[\hat{m}_i - \bar{m}_i \right] \geq \sigma_i \equiv \sqrt{C^{\mathrm{CR}}_{ii}} \tag{30}$$

where $\bar{m}$ is the true parameter and σ_i is the minimum variance for each parameter m_i.

This variance depends on the FIM for that parameter and the off diagonal for the remaining parameters. Thus if additional parameters are included in the search and they are coupled to the present parameter then the variance might change significantly. Thus increasing the number of parameters can lead to a large error.

Coupling between parameters tend to make the inversion problem more difficult, it is then preferable to design an approach and use a parameterization such that coupling is minimized. The coupling between the parameters can be studied by an eigenvalue decomposition of the covariance matrix. The eigenvector shows how a problem should be discretized so that the parameters become uncoupled. One problem with such an approach is that the eigenvectors change as a function of parameter values.

It assumed that source can be described as Gaussian stochastic process. for further discussion of derivation of the bounds see Ref. [51, 50]. For a Stochastic signal the Fisher Information matrix $\mathbf{F}$

$$\mathbf{F}_{ij}^{\text{sto}} = \sum_{\omega} \sum_{\text{p,v,h}} \operatorname{tr} \left[\mathbf{R}^{-1}(\bar{\mathbf{m}}) \frac{\partial \mathbf{R}(\bar{\mathbf{m}})}{\partial m_i} \mathbf{R}^{-1}(\bar{\mathbf{m}}) \frac{\partial \mathbf{R}(\bar{\mathbf{m}})}{\partial m_j} \right] \ . \tag{31}$$

Thus the Fisher information matrix is based on the assumed correlation matrix and its derivative at a given true point $\bar{\mathbf{m}}$. It is thus a local method. For analytic comparison of different estimators this formula is often expanded using particular assumptions about the signal. For our purpose this is not necessary as it is in its most general form and is ready for numerical work. For computing the derivative of the correlation matrix $\mathbf{R}$ the usual model for is assumed

$$\mathbf{R}(\omega) = \mathsf{E}\left[\mathbf{y}^{\dagger}\mathbf{y}\right] = \mathsf{E}\left[S^2\right]\mathbf{q}^{\dagger}\mathbf{q} + \mathbf{C}_{\text{n}}(\omega) \ , \tag{32}$$

where $\mathbf{C}_{\text{n}}(\omega)$ is the noise covariance matrix, which is assumed independent of the model parameters $\mathbf{m}$. and the derivative is then obtained.

$$\frac{\partial \mathbf{R}}{\partial m_i} \quad = \quad \mathsf{E}\left[S^2\right]\left[\frac{\partial \mathbf{q}}{\partial m_i}\mathbf{q}^{\dagger}(\mathbf{m}) + \mathbf{q}(\mathbf{m})\frac{\partial \mathbf{q}^{\dagger}}{\partial m_i}\right] \tag{33}$$

In the computations it is assumed $\mathsf{E}\left[S^2\right] = 1$. The derivatives of the field is computed using analytic gradients [18]

9

Shape functions

Here regularization is introduced via shape functions. These can be seen as a coordinate transformation h_{ij} between the input parameters m_i required for the forward modeling program and a more efficient set of parameters μ_j:

$$m_i = \sum_{j}^{M_s} \mathbf{h}_{ij}\mu_j, \tag{34}$$

where h_{ij} is the jth shape function or basis function, μ_j is the coefficient associated with this shape function and M_s is the number of shape functions used. The advantages of shape functions are:

- They constrain the solution to a certain class of expected profiles.
- They describe the variation of the parameters with fewer coefficients and in so doing reduce the number of unknowns. This can also constrain the solution to be more physically correct. For example, by linking the sound speed in the sediment together, we can describe them by a smoother function and, furthermore, reduce the number of unknowns.
- They link correlated parameters, resulting in better search performance.

Regularization using shape functions may decrease the correlation between parameters and this would improve the inversion results. This was done in [52] where we inverted for the slope and the offset of the water sound speed, instead of inverting for the absolute sound speed at discrete points.

For each of these files a "!" as the first character of a line means a comment— There should be no blank lines. The format is given in Table 2. The starting values of the shape function coefficients are given because we do not use the actual values in the forward-model-input block for the parameters described by shape functions.

9.0.1 Example of a shape function file

This example is based on the input file in Sect. 11.4.4, with SNAP as the forward model. An option **E** is then included in the option line of the input file. The example

Input parameter	Description
Block I: DIMENSIONS (1 line)	
Neof Neofvar Nblock	Neof: Number of shape functions Neofvar: Number of points to specify shape function Nblock: Number of blocks to specify shape functions
Block II: PARAMETERS (`Neofvar` lines)	
parm index . . .	Repeated Neofvar times. parm: Pointer to physical parameter. For key see Sect. 11. index: Addresses a specific variable in a vector (for SNAPRD, PROSIM, CPROSIM, RAMGEO and TPEM, `index` refers to 2 variables, `index` and `index2`)
Block III–Nblock+IV: SHAPE FUNCTIONS	
NeofBlock NeofvarBlock a1 a2 ... aNeofBlock ...	The coefficients – repeated NeofvarBlock times
Block Nblock+V: starting amplitudes (1 line)	
μ_1 μ_2 μ_3	μ_i is the starting amplitude for the shape function i

Table 2 *Shape function file structure.*

uses the same pointers as in SNAP, see Sect. 11.4.3.

Instead of the two sound speed points at the top and bottom of the water column (pointers 2-1 and 2-2), we will describe the water sound speed profile using two shape functions. The first shape function (11-1) is constant over depth, and the second shape function (11-2) describes the decrease in sound speed at the bottom. Hereby we have obtained a more efficient parameter set as the gradient is more important than the absolute offset. We will also reduce the number of free parameters in the bottom by binding the lower sediment speed (3-2) to the basement speed (12-1) using one shape function (11-3). In terms of coordinate mapping, Eq. (34), we express this as

$$\begin{bmatrix} \text{2-1} \\ \text{2-2} \\ \text{3-2} \\ \text{12-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \text{11-1} \\ \text{11-2} \\ \text{11-3} \end{bmatrix}. \tag{35}$$

Because it consists of two sub-matrices, we describe the mapping matrix using two blocks, `Nblock`=2. The use of sub-matrices is especially useful for large problems.

```
!Shape function example for sspmis
3 4 2                          ! Neof Neofvar Nblock
2 1                            ! upper speed in water
2 2                            ! Lower speed in water
3 2                            ! lower sediment speed
12 1                           ! basement speed
! The first block describe the water profile
2    2                         ! the size of first sub-matrix
1    0
1   -1
! The second block describe the bottom profile
1  2                           ! the size of second sub-matrix
1
1
1500 20 1750                   ! starting amplitudes
```

Also the optimization parameter block of the input file must be changed:

```
5                              ! nparm
11 1    1497.5 1502.5  51      ! upper sound-speed point
11 2     15     25     101     ! gradient of sound speed
11 3    1600   1800     51     ! basement speed
 9 1    5000   10000    51     ! source range
 8 1    0.01    100     51     ! source depth
```

10
Output files

The *.out is the default output file. For each execution of POST a summary of the estimated parameters is also added to the files **results** and **results.m** in the current directory. With minor modifications the **results.m** can be used with MATLAB to analyze the convergence of different inversion approaches.

The plot files (*.plt, *.plp, *.bdr and *.cdr) use the standard SACLANTCEN plotting programs, but as described in Sect. 5.1, they are better processed with MATLAB.

10.1 The *.mat, *b.mat and *.ext files

All three files use the same format:

```
do i=1,q
     write(10,*)i,ipop,fit(i),(model(jn,i),jn=1,nparm)
enddo
```

where the variable `model` contains an integer representation of the model vector for each individual `i`. In total there are `q` model vectors. It is quite easy to read these files into MATLAB and plot the results.

***.mat**: The file is written from SAGA. Each time a new forward model has been ran it is written out using pointers model vectors are written out to the *.mat file. There will be as many models as forward modeling runs.

***.ext**: The file is written from SAGA. Each time a new forward model has been ran it is written out using the physical dimensions.

***b.mat**: The file is written from POST. It contains a sorted list of all the individuals used in the post-processing.

11

Forward models and examples

A short description of the forward models that are currently incorporated into the SAGA program is given. This description requires familiarity with the particular forward code, which is strongly coupled to the program to ensure computational efficiency.

As each forward model is a subroutine it has some limitations relative to the stand-alone version. Firstly, it reflects the forward model at the time of porting. Secondly, the input subroutine is usually rewritten and therefore some options available in the original model might not be available in the SAGA version.

11.1 Forward model: OAST

The forward model for OAST follows *nearly* the format of the SAFARI manual [2]. We use the format described in the upgrade notes [3]. But it is based mainly on OASES Version 1.2. The source-receiver range can be read in using two formats. See the input examples.

In the transmission loss module of OASES the complex pressure is found by using the fast field approximation and then doing a Fourier transform from wavenumber domain to range domain. This has the disadvantage that the pressure is only computed at discrete intervals with a spacing of about one wavelength which causes an inaccurate determination of the phase. Therefore in SAGA full integration similar to that in the OASES pulse module is used.

It is difficult to determine the correct wavenumber sampling when using several frequencies and when the environment is changing during the optimization. Therefore automatic sampling has been introduced for the transmission loss module based on automatic sampling for the pulse program in OASES 2.0. Similarly to OASES it is activated by specifying a negative number of wavenumber samples. The minimum and maximum horizontal phase speeds (`Cmin` and `Cmax`) that should be used in the sampling are still important. The method used is based on the supplied `Cmin` and `Cmax`; it computes the minimum and maximum wavenumbers for the maximum frequency. For all frequencies the integrand is sampled in the range of these two wavenumbers. The number of required samples at each frequency can then be determined from geometric considerations.

It is possible to use more than one physical quantity in one inversion. Thus any combination of pressure, vertical velocity and horizontal velocity can be used. The velocities are normalized with $\rho_{\mathrm{ref}} c_{\mathrm{ref}}$ in order to have same dimensions and magnitude as the pressure ($\rho_{\mathrm{ref}} = 1000$ kg/m^3 and $c_{\mathrm{ref}} = 1500$ m/s. Specify N (pressure), V (vertical velocity) and/or H (horizontal velocity) in the oases option line. SAGA always works with the physical quantities in the above order. At the moment, when more physical quantities is observed the frequencies has to be specified in multiples of number of field type.

11.1.1 *Change in input format*

- Frequency line: The OASES standard is to specify:
  `Fmin Fmax Nfreq`.
  Hereby `Nfreq` frequencies are used from `Fmin` to `Fmax`. If `Nfreq` is negative then | `Nfreq` | frequencies are read individually in the next line:
  `Fdum Fdum -Nfreq`
  `freq(1) freq(2) ... freq(Nfreq)`
- Receiver line: The OASES standard is to specify:
  `RDmin RDmax Nrd`.
  Hereby `Nrd` receiver depths are used from `RDmin` to `RDmax`. If `Nrd` is negative then | `Nrd` | receivers are read individually in the next line:
  `Rdum Rdum -Nrd`
  `rd(1) rd(2) ... rd(Nrd)`
- For options **d**, **f** and **c** the *range format* lines are:
  `no_of_ranges`
  `range(1), ... range(no_of_ranges)`
  hence each used range must be specified. For option **D** the format is:
  `R_min R_max`
  thus only ranges between `R_min` and `R_max` are used in the input file.
- If the shear speed is -999.999 then the layer is an acoustic gradient layer. The top sound speed corresponds to the P sound speed in the layer and the bottom sound speed to the P sound speed in the layer below.

11.1.2 *Additional OAST options*

i Incoherent averaging of transmission loss. At range r it is found by averaging the transmission loss in range from $r - 0.116r$ to $r + 0.116r$ as described in Ref. [53].

t Tilt of receiver array. It is measured as the horizontal deviation at the last receiver measured in meters. When this option is specified, it is the fourth parameter in the receiver-depth line of the input file. It works only for a single range.

A horizontal array can be simulated using this option. First specify the minimum and maximum receiver depth of the array. The tilt is relative to the full length of the array. The source-receiver distance is the distance between the source and the first receiver. By specifying a horizontal array in this way, the same options as for a vertical array can be used.

11.1.3 Pointers in OAST

The pointers `parm` and `index` are used to map between the optimization variable and the environmental parameter to be optimized. The first pointer `parm` points to the type of parameter and the second `index` identifies the actual parameter within that parameter type. For some parameters `index` is not important (source depth). The pointer `parm` can take the following values:

- **1** Depth of layer `index`. It should be between the depth of layer `index-1` and `index+1`.
- **2** P-speed of layer `index`.
- **3** S-speed of layer `index`.
- **4** P-attenuation of layer `index`.
- **5** S-attenuation of layer `index`.
- **6** Density of layer `index`.
- **7** Thickness of layer `index`. The thickness of one layer is changed by opening up the corresponding layer and keeping all other thicknesses constant; i.e. the depth below the corresponding layer will change.
- **8** Source depth.
- **9** Source-receiver range.
- **11** Shape function coefficient; `index` points to the shape function number.

11.1.4 OAST example. File: **layer11**

This example has been used in a published paper [16]. The options (line two in the input below) specify that we are using wavenumber domain option **w**, reading data using format option **d**, writing out the data from the starting model to the *.obs file option **W**, we are only matching the magnitude of the observations option **G**, using objective function option **N**, plotting the measured data and the best inverted results option **p**, and the baseline environment are superimposed on the *a posteriori* distribution plot option **A**.

```
11 layers- example used in GA-paper.
w W d G N p A                          ! Options
1000 32   1                      ! niter, q, npop
0.8 0.5 0.05                     !px, pu, pm
N J I T                          ! OASES options
100 100 1  0                     ! Fmin Fmax Nfreq
13                               ! Number of layers
0      0   0  0   0   0          ! First layer is vacuum
0   1500   0  0   0   1
50  1600   0 0.1 0.2 1.6
60  1600   0 0.1 0.2 1.6
70  1600   0 0.1 0.2 1.6
80  1600   0 0.1 0.2 1.6
90  1600   0 0.1 0.2 1.6
100 1800   0 0.1 0.2 2.0
110 1800   0 0.1 0.2 2.0
120 1800   0 0.1 0.2 2.0
130 1800   0 0.1 0.2 2.0
140 1800   0 0.1 0.2 2.0
150 2200   0 0.1 0.2 2.2
50                               ! source depth
50 50 1                          ! first, last, number of receivers
1400 3000                        ! Cmin, cmax
64 1 64                          ! wavenumbers

1                                ! The Range information is dummy
1300                             ! For wave number inversion

11                               ! nparm q
2 3  1500 2300 128            ! P-speed, lay 3
2 4  1500 2300 128            ! P-speed, lay 4
2 5  1500 2300 128            ! P-speed, lay 5
2 6  1500 2300 128            ! P-speed, lay 6
2 7  1500 2300 128            ! P-speed, lay 7
2 8  1500 2300 128            ! P-speed, lay 8
2 9  1500 2300 128            ! P-speed, lay 9
2 10 1500 2300 128            ! P-speed, lay 10
2 11 1500 2300 128            ! P-speed, lay 11
2 12 1500 2300 128            ! P-speed, lay 12
2 13 1500 2300 128            ! P-speed, lay 13
```

First the observed data should be generated. This is done by first deleting the *.in file. Then run SAGA using the command:

```
saga layer11 oast
```

It will write the observed data to the *.obs file. This file is then copied to the *.in file. Now, run SAGA again. The output of SAGA is the following:

```
 best of all energy  3.2385509E-02
best-of all      deviation
 1  1588.1          -11.8
 2  1903.1          303.1
 3  1638.5           38.5
 4  1915.7          315.7
 5  1991.3          391.3
 6  2161.4          361.4
 7  1966.1          166.1
 8  2287.4          487.4
 9  2066.9          266.9
10  2060.6          260.6
11  2041.7         -158.2
```

It is seen that in the first layer, which is the most important one, the sound speed is estimated quite well. Here only one population was used and this is not a sufficiently large sample for statistics of the estimates. We therefore increase the number of populations to 50 and run SAGA again. When SAGA has finished, the post-processor is executed:

```
post layer11 oast
```

The result of the run is then

```
best fit (best, ppd, mean)  1.6866000E-03  2.7032106E-03  3.2024365E-02

                             best-of all  most likely    mean   std-dev
 1 P sound speed (m/s)   3      1594.4      1594.4     1600.0     0.011
 2 P sound speed (m/s)   4      1607.0      1607.0     1607.0     0.019
 3 P sound speed (m/s)   5      1588.1      1581.8     1616.0     0.044
 4 P sound speed (m/s)   6      1619.6      1619.6     1652.1     0.057
 5 P sound speed (m/s)   7      1607.0      1600.7     1766.6     0.188
 6 P sound speed (m/s)   8      1814.9      1814.9     1782.2     0.144
 7 P sound speed (m/s)   9      1751.9      1751.9     1788.6     0.157
 8 P sound speed (m/s)  10      1733.0      1777.1     1764.1     0.154
 9 P sound speed (m/s)  11      1537.7      1613.3     1770.1     0.281
10 P sound speed (m/s)  12      2048.0      1947.2     1937.9     0.195
11 P sound speed (m/s)  13      2073.2      2098.4     1971.1     0.325
```

The first line [best fit...] indicates the best value of the objective function obtained using the best, most likely and mean model vector. The value of the model vector then follows for each of the estimates. The output is explained in Sect. 3.

From these results it is observed that for the first four layers we have quite good results. The normalized standard deviation is small. For the lower layers, the sound speeds are not so well determined, the standard deviation is also quite high. It is natural that the sound speeds in the lower layers are less well determined, as they have less influence on wave propagation in water.

Next, the results are plotted using MATLAB:

```
>> plotsaga
SAGA filename ? layer11
```

The *a posteriori* distributions for this case are shown in Fig. 7. When the distributions cluster around a certain value, it is an indication that a parameter is well determined. The above plot was produced by weighting each model vector according to its fit, as better fits are more likely to represent the true solution, see Sect. 2.2. Alternatively, we could use a uniform weighting of the sampled model vectors, Fig. 8. This can be done by entering option **u** in the SAGA option line. The distributions now become flatter and less centered around the true values.

The best obtained fit is shown in Fig. 9.

Using the information in the **layer11b.mat** file it is also possible to construct a plot of the most likely model vector. An example of that is given in Fig. 7 of Ref. [16]. It is also possible to plot the correlation between two parameters as presented in Fig. 8 of Ref. [16].

11.1.5 OAST example. File: **tellaro_oast**

This example shows an inversion for observed transmission loss data. The measured data was taken at the Tellaro site, in the Gulf of La Spezia. The input file for this run was used in Refs. [18, 54], for further details see these papers. This inversion using OASES is similar to the example with SNAP in Sect. 11.4.5. For this type of long range propagation SNAP is preferable, both because the CPU-time is less and because this code is easier to use. The additional information about shear and attenuation that can be obtained using OASES is not important in the present example.

11.2 Forward model: OASR

OASR is the reflection coefficient module of SAFARI. The forward model for OASR follows the format of the SAFARI manual [2, 3].

Some numerical codes model the bottom using the reflection loss as a function of angle of incidence, whereas others use the physical sound speed profile in the bottom.

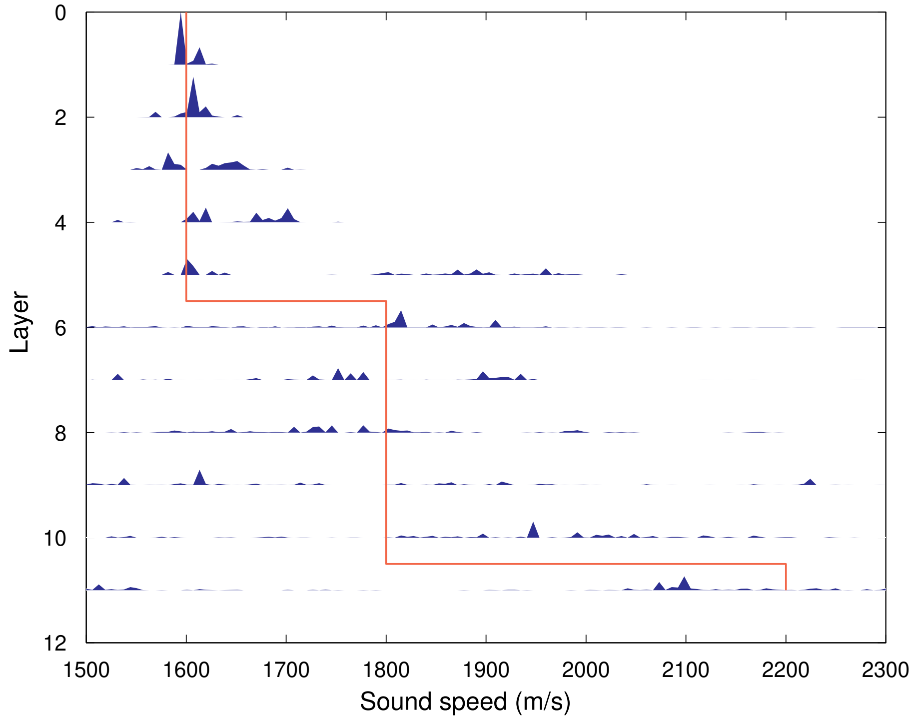

Figure 7 *The a posteriori distributions for* **layer11**. *The baseline environment that generated the data is shown in red.*

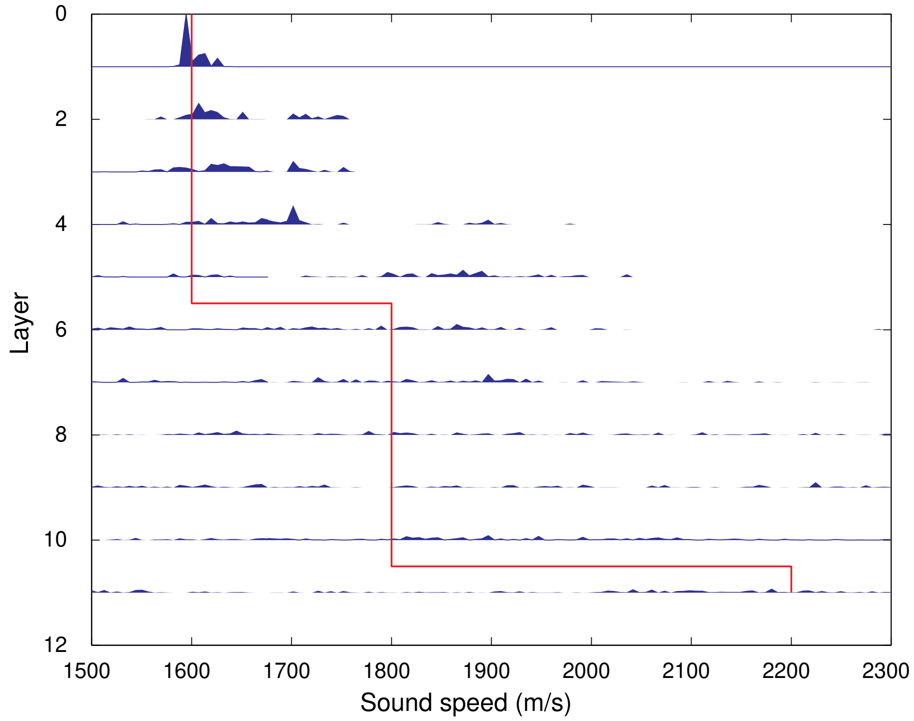

Figure 8 *The uniform a posteriori distributions for* **layer11**. *This represent a histogram of the model vectors obtained from the inversion. The baseline environment that generated the data is shown in red.*

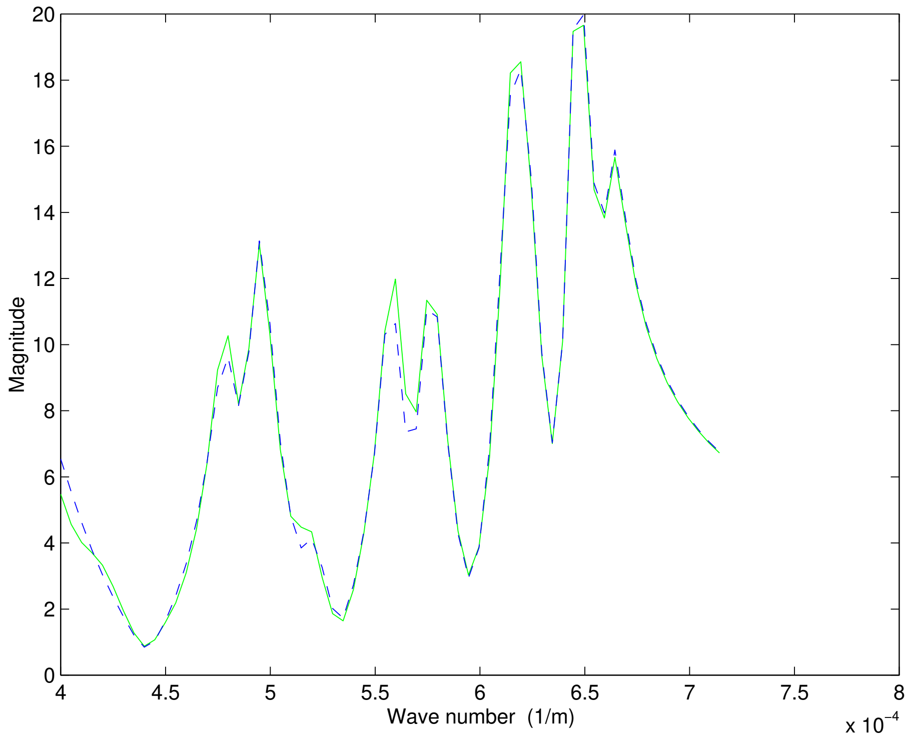

Figure 9 *The fit in wavenumber space for* **layer11**. *The green line is the observed data, and the blue line is the computed data with the best set of parameters.*

The SAGA-OASR module can be used to estimate a physical bottom from a given reflection loss curve.

11.2.1 Pointers in OASR

The pointers are the same as for OAST, see Sect. 11.1.3, except that the pointers to source and receiver geometry are not used.

11.2.2 OASR example. File: **rfl**

This synthetic example demonstrates the use of the OASR module. From the option line it is seen that the data is read using format option **d**, the calculated data from the starting model is written to the *.obs file option **W**, using objective function option **N**, and the observed and calculated data are then plotted with option **p**.

```
 Refection coefficient.
d W N p                                  ! options
1000 32  2                               ! niter, q, npop
0.8 0.5 0.05                             ! px, pu, pm

3
0   1500    0  0   0   1  0
0   2000    0 0.1 0.2 1.6 0
30  2500    0 0.1 0.2 1.6 0
50 100 2                                 ! frequencies: Fmin Fmax Nfreq
0 90 100                                 ! angle

2  ! nparm
2 2 1500 3093.75 999
2 3 1500 3093.75 999
```

The result of the inversion is quite good, as shown by the output from SAGA:

```
   best of all energy  7.1087794E-07
  best-of all      deviation
   1  1999.843           -0.157
   2  2499.687           -0.313
```

Both sound speeds in the bottom are well determined. In this case there are so few variables that it does not seem necessary to display the *a posteriori* probability distributions.

11.3 Forward model: OASTG

This version should only be used if gradients of the parameter vector are used in the optimization [18]. At present it works with only one frequency. It is more complicated to use than the standard OAST module.

This forward model is based on OAST. The input file and the pointers are therefore identical to the ones described in Sect. 11.1. If gradients are not used it has the same functionality as OAST, but it uses more dynamic memory.

The program works with "exact" derivatives from OASES with acoustic layers, thus all optimization parameters should be in acoustic layers. The other layers can be of any type (e.g. elastic).

This program has great potential for problems that require computation of gradients, as for example Cramer-Rao bounds. When gradients are computed based on a finite difference approach there are generally large errors involved in the computation, as opposed to the present exact approach.

11.3.1 OASTG example. File: **horiz**

This illustrates the computation of the Cramer-Rao bound for a horizontal array [55]. The horizontal array is 150 m long and the source is at 10 m depth and 1000 m range. The water column is 50 m deep with a sound speed of 1500 m/s and the sediment speed is 1600 m/s. Noise with a SNR=10 db is added to the covariance matrix, option **z** in the SAGA option line. Gradients, option **g** and covariance matrix option **c** are used. The ambiguity function, Fig. 10a shows that the source coordinates are hard to find, the maximum of the ambiguity surface is 0 db and the dynamic range is 1.5 dB. Figure 10b and 10c show the resolution as estimated from the Cramer-Rao bound, based on Eq. (31). The resolution is the square root of the diagonal entries in the Cramer-Rao covariance matrix. It is seen from the figure that the horizontal source range is less well resolved than the depth.

11.4 Forward model: SNAP

If the wave propagation is long-range it is practical to model the wavefield using normal modes. This is much faster than the more precise method of wavenumber integration.

Also practice as well as theory has shown that good results can be obtained using a range independent environment even in a range independent environment. Though

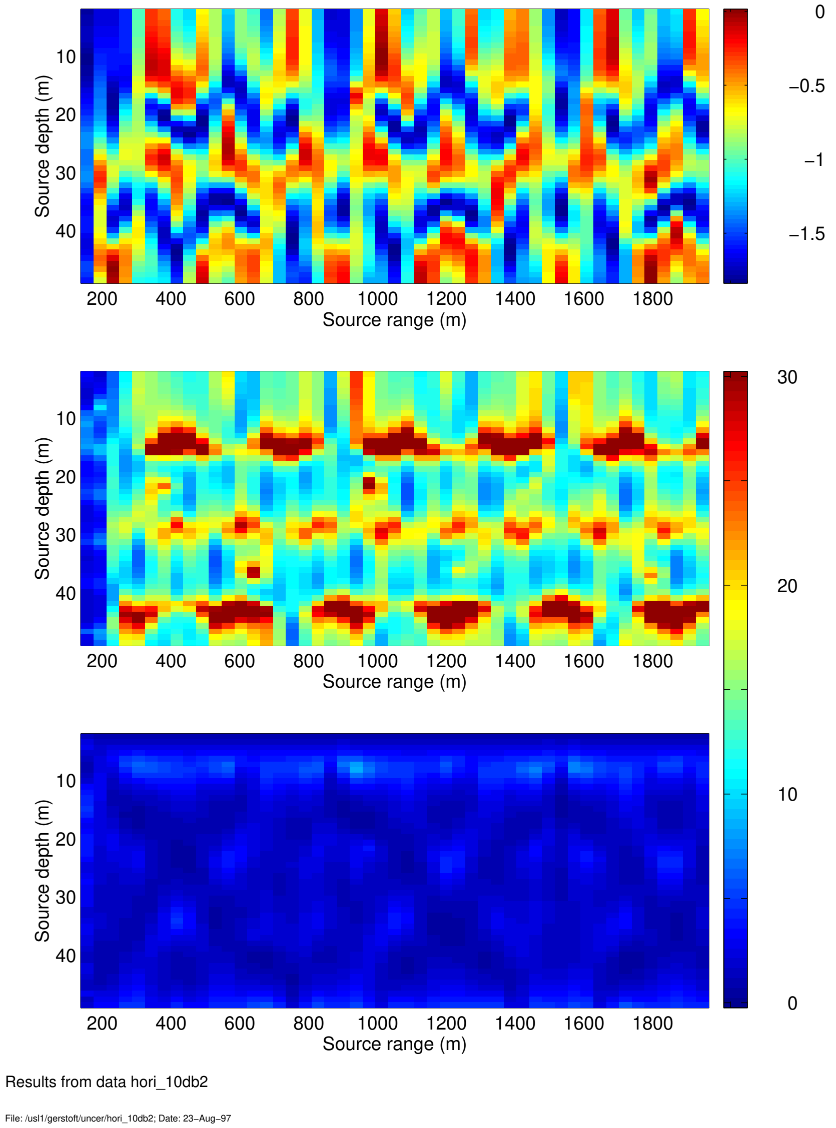

Figure 10 *The resolution when estimating the main wave guide parameters for the* **horiz** *case. The pressure on a horizontal array with 20 phones and a length of 150 m is used. a) ambiguity function, b) source range resolution in [m], c) source depth resolution in [m].*

some of the parameter will be offset if a range independent model is used in a range dependent environment [59].

The main part of the input follows the description in the SNAP manual [1]. The wave attenuation in water should be specified (in the water-depth line of the input file). If this attenuation is zero, the default attenuation is used, as in the original SNAP model.

The frequencies that are used in SNAP are specified by either:

```
Nfreq Nmodes                !
freq(1) freq(2)...  freq(Nfreq)
```
or
```
-1 Nmodes
Dfreq Fmin Fmax
```

For the optimal environment a transfer function is generated. When computing transfer functions with the optimal environment it is possible to change the frequency sampling (the _last) for the broadband frequency input mode as follows: `-1 Nmodes Dfreq Fmin Fmax Dfreq_last Fmin_last Fmax_last`
The transfer function is written in OASES format (file *.trf) and can then be read by the OASES postprocessor or similar.

11.4.1 Change in input format

- Receiver line: The SNAP standard is to specify:
  `RDmin RDmax Nrd`.
  Hereby `Nrd` receiver depths are used from `RDmin` to `RDmax`. If `Nrd` is negative then | `Nrd` | receivers are read individually in the next line:
  `Rdum Rdum -Nrd`
  `rd(1) rd(2) ...  rd(Nrd)`
- For options **d**, **f** and **c** the *range format* lines are:
  `no_of_ranges`
  `range(1), ...  range(no_of_ranges)`
  hence, each used range must be specified.
  If `no_of_ranges` are negative then the format is
  `no_of_ranges`
  `R_min R_max`
  This will create `no_of_ranges` equidistantly spaced ranges between `R_min` and `R_max`.

  For option **D** the format is:

`R_min R_max`
thus only ranges between `R_min` and `R_max` are used in the input file.

11.4.2 Options in SNAP

In the input file an option line is introduced for SNAP. Currently, there are only two options:

- **i** Incoherent addition of modes.

- **t** Tilt of receiver array (both SNAP and SNAPRD). It is measured as the horizontal deviation at the last receiver measured in meters. When this option is specified, it is the fourth parameter in the receiver-depth line of the input file.

  A horizontal array can be simulated using this option. First specify the minimum and maximum receiver depth of the array. The tilt is relative to the full length of the array. The source-receiver distance is the distance between the source and the first receiver. By specifying a horizontal array in this way, the same options as for a vertical array can be used. with multiple sources it is not possible to perturb the source depth.

- **m** Multiple sources. It is assumed that all sources has unit strength. The sources are on a vertical array. The response of all sources in summed for each receiver. With multiple sources it is not possible to perturb the source depth.

  For option **m** the source line in the input file should contain:

  `sd sdlow nsrd !  upper lower and number of sources.`

  If nsrd is negative then —nsrd— individual source depths are read in the next input line.

- **M** Multiple source. The input file should be similar to the **m** option. In addition also the complex source spectrum should be supplied in the *.sou file. It has the same format as the "vertical array data" input file described in sect. 7.4.

- **h** Bathymetry approximation done by stretching the modes. This option works only when all the other parameters are geometric and it is then possible to avoid recomputing the modes. The modes are computed for the reference environment and for subsequent changes in bathymetry the mode shapes are stretched according to the relative change in bathymetry.

- **s** Scaling factor on the pressure for each frequency. This could be used to change the weighting for each frequency used in for example a transmission loss inversion. This is specially if the source spectrum is not well known. For each frequency the scaling factor is default 0 dB. Option **n** should then be used.

- **T** multi-static modeling. From the source the signal is transmitted to a perfect echo-repeater and from there the signal is then sent to the receivers. The depth of the echo-repeater `ed` and source-repeater range `er` to the receiver must be specified in a line between the source-line and receiver-line.
```
sd                       !  source depth
ed er                    !  echo-repeater depth and range
```
The input file `multsamp` (in the examples directory) is an example of how to use this option.

- **p** Mean grain size inversion. Instead of of inverting for sound speeds, densities and attenuation we just invert for the mean grain size in the sediment all sound speed, densities and attenuation are computed using the Hamilton-Bachman relations[76, 77]

11.4.3 Pointers in SNAP

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not important (source depth). The pointer `parm` can take the following values:

1 Water depth. The depth is changed by moving the last point in the water.
2 Sound speed in water; `index` refers to the actual point.
3 Sound speed in sediment; `index` refers to the actual point.
4 Attenuation. Depending on the value of `index` it refers to the following attenuation: `index` = 0(water), 1(sediment), 2(bottom), 3(shear bottom).
5 Surface roughness, `index` = 1(ocean surface), 2(bottom surface).
6 Sediment density.
8 Source depth.
9 Source-receiver range.
11 Shape-function coefficient; `index` points to the number.
12 Bottom P-speed.
13 Bottom S-speed. The shear in SNAP should have a low value ($<$ 500 m/s), otherwise the perturbation correction is not valid.
14 Bottom density.
15 First receiver depth. The spacing is constant, thus it is a vertical translation.
16 Depth of speed point in water; `index` refers to the actual point.
17 Sediment thickness. The thickness is changed by moving the last point in the sediment.
18 Depth of speed point in sediment; `index` refers to the actual point.

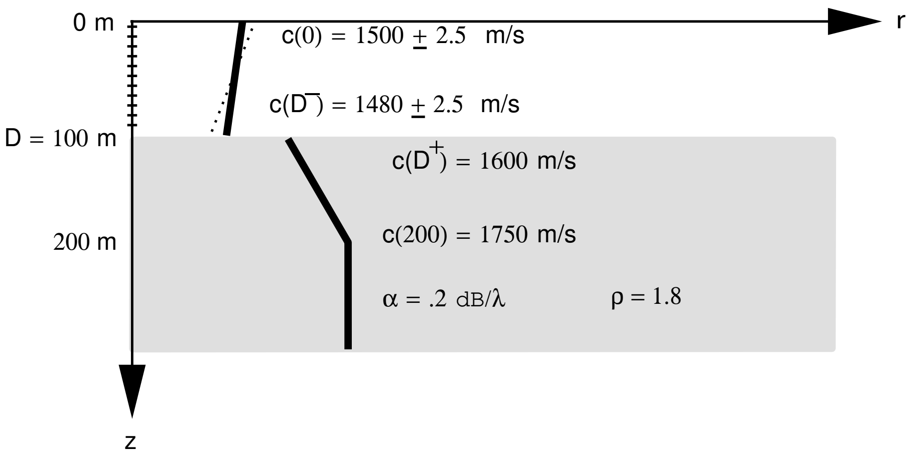

Figure 11 *The environment for the* **sspmisa** *case. The source coordinates are (9.3 km, 78 m) for SNR = 40 dB.*

19 Tilt. SNAP option **t** must be specified.
21 Array shape. See the SNAP3D, Section . 12.
22 Source strength. Used For SNAP-option **s**. *index* refers to each frequency.
23 Depth of echo-repeater (*index=1*) and source echo-repeater range (*index=2*). SNAP option **T** must be specified.

11.4.4 SNAP example. File: **sspmisa**

This example is from the 1993 Matched Field Workshop [46], and has been used in a published paper [52]. The environment is shown in Fig. 11. The observed data are based on a synthetic data set from a normal mode code using a 250 Hz source in shallow water. The data are received on the 20 hydrophones spanning the whole water column. White Gaussian noise was added to the data vectors to obtain a SNR = 40 dB [46]. Only four parameters are unknown in this case; the source range and depth and the ocean sound speed at the top and the bottom. Each of the parameters can take 51 discrete values.

The options in the input data file indicate that we are using a covariance matrix, option **c**, and that this matrix is scaled to a maximum Bartlett power of one, option **b**. We plot the variation of power across the array, option **p**. Option **t** in the SNAP-option line indicates that the array can be tilted; the fourth parameter in the receiver line defines the initial tilt. We use GA with 10 populations, each with 1000 individuals; 10,000 forward models are used in total.

```
sspmis_a   from the matched field workshop 93
```

```
c b  p                        ! options
1000  32  10                  ! niter, q,  npop
0.8 0.5 0.05                  ! px, pu, pm
t                             ! snap options
1  100                        ! No of freq, max modes
250.                          ! freq
100  0  0  0                  ! wd scat(1) scat(2) beta
  0 1499.4
  100 1481.6
100 1.8 0.2                   ! Sediment depth density and attenuation
  0 1600
  100 1750
1.8 0.2 1750                  ! bottom density, attenuation and sound speed
0   0                         ! bottom shear attenuation and sound speed

78                            ! source depth
5 100 20 0                    ! first & last receiver, # receivers, tilt

1                             ! number of ranges
9300                          ! receiver range

4                             ! nparm
2 1    1497.5 1502.5 51       ! upper sound speed point
2 2    1477.5 1482.5 51       ! lower sound speed point
9 1    5000   10000  51       ! source range
8 1    0.01    100   51       ! source depth
```

After running SAGA and POST we obtain the results:

```
best fit (best, ppd, mean)  3.4361999E-04  2.1212101E-03  5.5390596E-04
                                best-of all  most likely     mean   std-dev
1 Top Water sound speed (m/s)    1  1499.7      1498.6     1499.12    0.117
2 Bottom Water sound speed (m/s) 2  1481.9      1480.8     1481.33    0.116
3 Source range (m)               1  9300.0      9300.0     9300.0     0.000
4 Source depth (m)               1    78.0        78.0       78.0     0.000
```

The first line [best fit...] indicates the value of the objective function that is obtained using the best, most likely and mean model vector. The value of the model vector then follows for each of the estimates. The output is explained in Sect. 3.

It is seen that the source coordinates are perfectly determined. The sound speeds are also quite well determined, but the normalized standard deviation is much larger. The same can be concluded from plotting the 1-D marginal *a posteriori* distributions (Fig. 12). By examining the 2-D marginal *a posteriori* distributions (Fig. 13) this can be explained.. The reason for the poor resolution of the parameters is due to a strong correlation between the ocean sound speed at the top and at the bottom. The

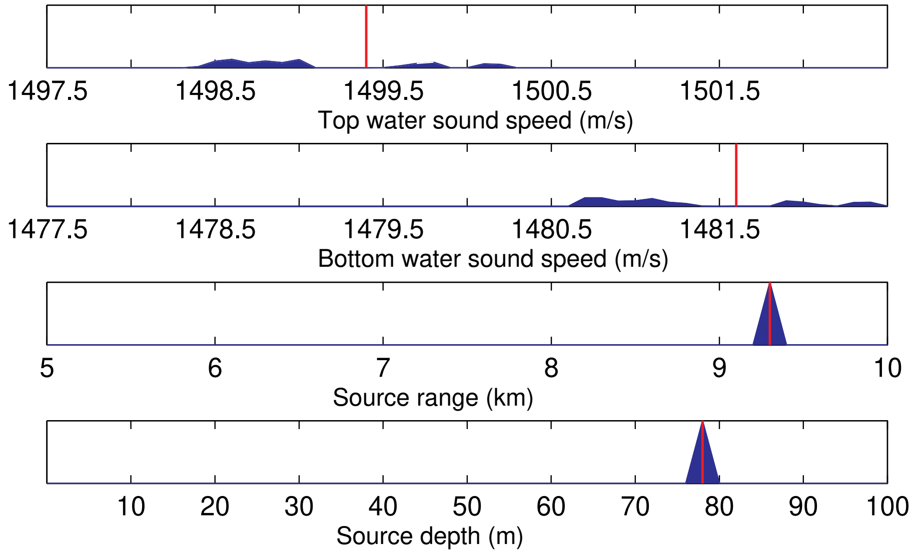

Figure 12 *The 1-D a posteriori distributions for the* **sspmisa** *case. The red line indicates the true value.*

2-D marginal *a posteriori* distribution was produced by introducing option **m12** on the option-line in the *.dat file and then running POST. The contour plot has been plotted using CPLOT, see Sect. 5.1.

The correlation of the two sound speeds can also be seen by fixing the source coordinates and then plotting a contour of the variation in the objective function for the two sound speeds, see Fig. 14. The ambiguity plot has been produced by putting option **C** in the option-line in the *.dat file; the number of discrete points for the two parameters has been changed to 40. It is then necessary to re-run SAGA, as this is seen as a 2-D exhaustive search. The contour plot has been plotted using CPLOT, see Sect. 5.1.

11.4.5 SNAP example. File: **tellaro_snap**

This example shows an inversion for transmission loss data as described in Refs. [18, 54]. The input has been simplified relative to the examples in [18, 54]. The options in the input file show that we are reading data using the range format, option **D**. We are only using the magnitude of the pressure and the data is weighted with $1/\sqrt{r}$ to put more importance on the shorter ranges, option **R**. The objective function is based on least squares, option **n**. We are plotting observed and synthetic data which best matches the observed data, option **p**.

```
tellaro-data
```

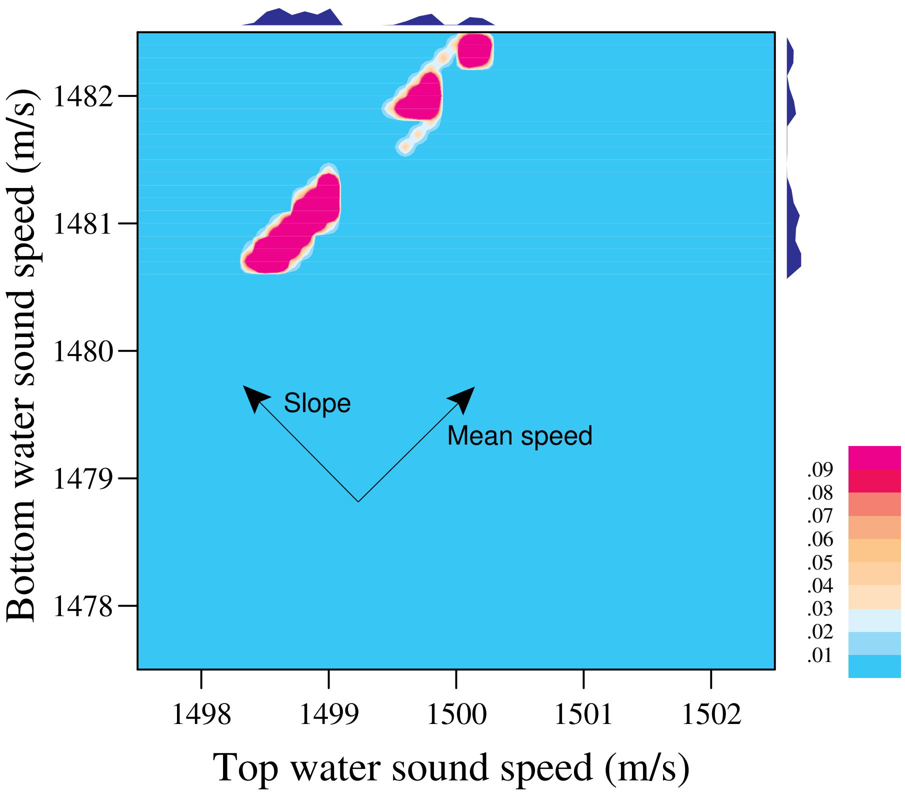

Figure 13 *The 2-D marginal a posteriori distribution between the upper and lower ocean sound speed for the* **sspmisa** *case. It is based on the same data as Fig. 12. The marginal 1-D distribution for each parameter are displayed on the top and to the right. By re-parameterizing the sound speed as slope and mean, a better resolution is obtained.*

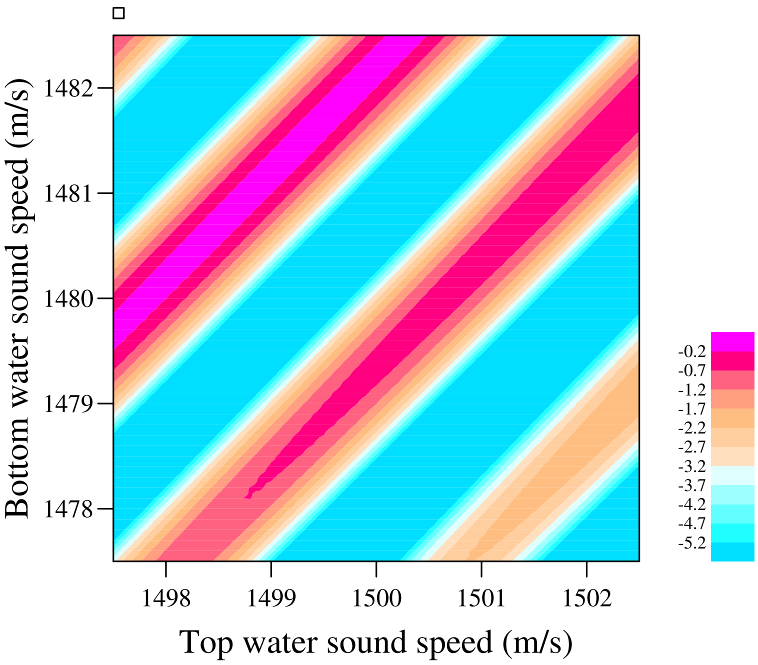

Figure 14 *Ambiguity surface for the* **sspmisa** *case.*

```
D n R p                 ! options
3000 64 15               ! niter, q, npop
0.8 0.5 0.05             ! px, pu, pm
                         ! option
1                        ! No of freq
330.                     ! the freq
15.0 0.  0. 1e-8         ! water-depth, roughness surf & bottom, small att.
  0   1523
 15   1523
24.0   1.7 0.25          ! sediment-depth, density, cp-attenuation
  0.  2000.              ! sediment sound speed
  3.  2000.              ! sediment sound speed
  9.  2000.              ! sediment sound speed
 24.  2000.              ! sediment sound speed
1.7 .25 2000
0. 0.                    ! No shear

7                        ! source depth
5. 5. 1                  ! NRD  RD(1:NRD)
270 1400                 ! for option D: Rmin, Rmax

6                        ! nparm
3 1 1480  1580 128       ! sediment speed 1
3 2 1480  1580 128       ! sediment speed 2
3 3 1480  1580 128       ! sediment speed 3
3 1 1480  1580 128       ! sediment speed 4
8 1 6.0   8.0  128       ! source depth
4 1 0.    0.5  128       ! attenuation in sediment
```

11.4.6 SNAP example. File: **ys3**

This example was used in the analysis of the Yellow Shark experimental data [56]. The source signal was transmitting 7 frequencies simultaneously in the band from 200–800 Hz. First the covariance matrix was estimated by averaging over several time snapshots using a multiple window technique [57, 58], as implemented in the MATLAB files. In the example the data at all frequencies are used simultaneously, but single frequencies could also be used. For results and discussion, see Ref. [56].

11.5 Forward model: SNAPRD

The range dependent version of SNAP allows for a more realistic description of the ocean environment. However, the inversion problem becomes more complicated. Both the CPU-time and the number of parameters required to describe the problem increase linearly with the number of range sectors.

The main part of the input follows the description in the SNAP manual [1]. The wave attenuation in water should be specified (in the water-depth line of the input file). If this attenuation is zero, the default attenuation is used, as in the original SNAP model.

11.5.1 Options in SNAPRD

For SNAPRD the receiver depths and ranges should be sampled equidistantly. In the input file an option line is introduced for SNAPRD. Currently, there is only one option:

i Incoherent addition of modes.

t Tilt measured as the horizontal deviation at the last receiver measured in meters. When this option is specified, it is the fourth parameter in the receiver-depth line of the input file. It works only for one range.

A horizontal array can be simulated using this option. First specify the minimum and maximum receiver depth of the array. The tilt is relative to the full length of the array. The source-receiver distance is the distance between the source and the first receiver. By specifying a horizontal array in this way, the same options as for a vertical array can be used.

11.5.2 Pointers in SNAPRD

Due to the increased dimensionality of the problem, we need 3 pointers to specify which parameter we are inverting. The two first pointers `parm` and `index` are the same as in SNAP, while the third pointer `index2` indicates the range sector.

11.5.3 SNAPRD example. File: **elbard**

This example is based on the data described in [47, 59]. Here we invert using a range dependent environment. The sound speed profile in the water is determined from an XBT at both the source and the receiver range. We use shape functions option **E** to produce a single range independent bottom and to describe the bottom sound speed more efficiently. The sound speed profile is modeled using the sound speed at the bottom interface and subsequent sound speed points are modeled using the increase in sound speed from the previous point.

We are using observations at 3 frequencies and adding them incoherently. The data is in covariance format, option **c**, and the covariance matrix was estimated by averaging over several time snapshots using a multiple window technique [57, 58] as implemented in the MATLAB files. We are plotting the variation of power across the array for each frequency, option **p**. The input file is:

```
 North elba
 E c p                   ! options
  2000  32  10           ! niter, q, npop
  0.8 0.5 0.05           ! px, pu, pm
                         ! snap options
 3 100                   ! No of freq,  max_no_modes
164.795 169.92   175.048
2                        ! Number of sectors
6                        ! Sector length in km
 129.9,0.,0.,0           ! water depth scatt(1),scatt(2),beta(0)
    0.     1525.5
   56.8    1526.4
   65.1    1523.1
   68.9    1520.6
   71.3    1517.2
   75.6    1512.1
   76.8    1510.9
   79.9    1510.1
  100.4    1508.4
  129.9    1508.3
 10 1.75 0.13            ! sediment thickness, r1, beta(1)
    0      1520.
    5      1580.
   10      1600
 1.8, 0.15, 1600.        ! bottom r2 beta(2),c2
 0.,0.                   ! bottom shear beta(3) c2s

6                        ! sector length  SECTOR 2
 127.1,0.,0.,0           ! water depth scatt(1),scatt(2),beta(0)
    0.0    1525.5
   56.4    1526.3
   60.1    1525.8
   66.6    1522.1
   70.6    1517.7
   77.9    1510.3
   86.1    1509.1
  101.1    1508.3
  127.1    1508.3
 10 1.75 0.13            ! sediment thickness, r1, beta(1)
    0      1520.
    5      1580.
   10      1600
 1.8, 0.15, 1600.        ! bottom r2 beta(2),c2
 0.,0.                   ! bottom shear beta(3) c2s

 80                      ! source depth
```

```
 112.7 18.7 48 1          ! rec depth
 1                        ! number of ranges
 5600                     ! receiver range

 7                        ! nparm
 9 1 1 5300 5900   128    ! source range
 8 1 1   70   85   128    ! source depth
 1 1 1  127  132   128    ! water depth
 1 1 2  127  132   128    ! water depth
11 1 1 1510 1560   128    ! sound speed sed
11 2 1    0  100   128    ! sound speed sed
11 3 1    0  100   128    ! sound speed bot
```

The shape function file is:

```
! shape function file for the North Elba range dependent case
3 8  1            ! No. of shape functions, No. of points, No of blocks
3 1  1            ! sound speed sed
3 1  2            ! sound speed sed
3 2  1            ! sound speed sed
3 2  2            ! sound speed sed
3 3  1            ! sound speed sed
3 3  2            ! sound speed sed
12 1 1            ! sound speed basement
12 1 2            ! sound speed basement
! Block 1: coupling of sed 1
3 8
1 0 0
1 0 0
1 1 0
1 1 0
1 1 1
1 1 1
1 1 1
1 1 1
1520 20 30        ! starting values
```

For results and discussion, see Refs. [47, 59].

11.5.4 SNAPRD example. File: **shot05**

This example is based on the shot data described in [60, 61]. Here we invert using a range dependent environment. The sound speed profile in the water is determined from an XBT at both the source and the receiver range.

The stability of the parameter estimate depends on the time-frequency variability of the energy distribution in the received data. To demonstrate this variability, the array-averaged spectrogram is calculated for a 256 ms time window which slides over the time domain with a 10 ms increment (Fig. 15). It can be seen that the energy content remains at the same level for only a short period (from 250 to 450 ms) corresponding to the high energy segment of the signal. Parameter estimates based on this time segment are expected to be very stable, and it is used here to estimate the covariance matrix for the frequencies used.

The data is processed using two objective functions, the default multi-frequency objective function, option **c** (unnormalized), and one where the covariance matrix for each frequency is normalized by dividing with the sum of the diagonal, option **b**

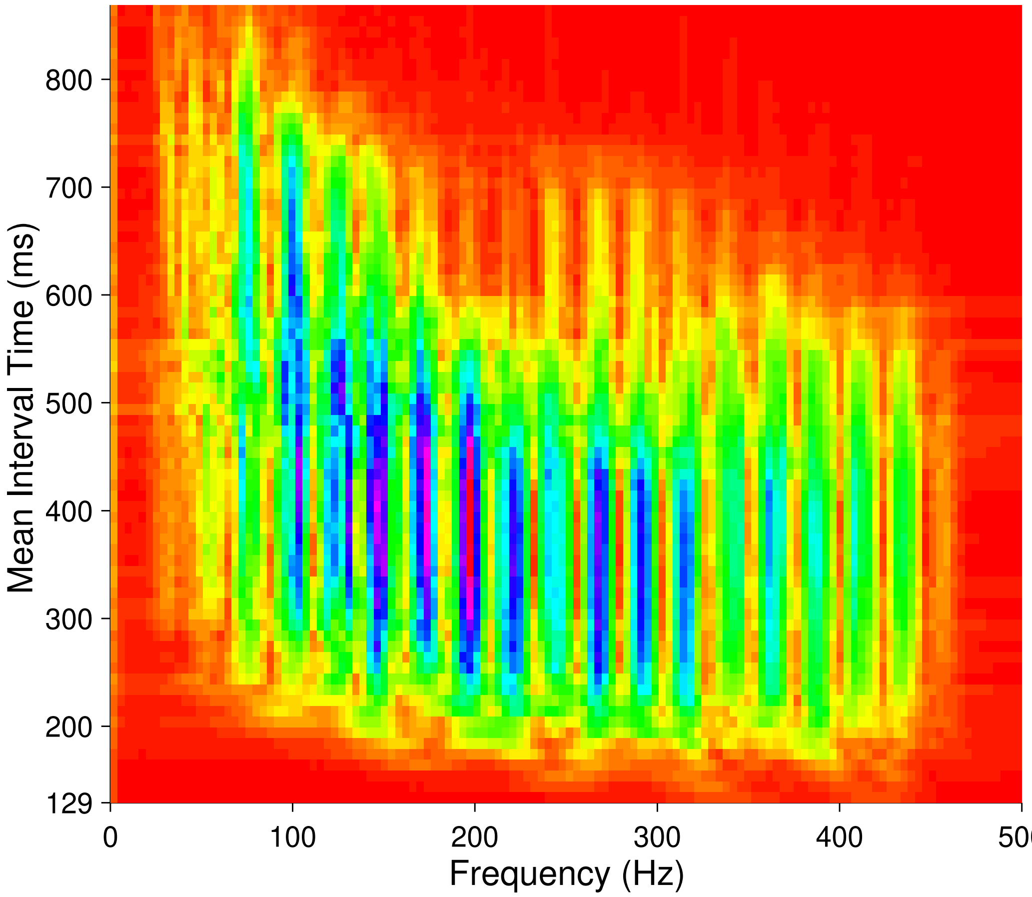

Figure 15 *Array-averaged spectrogram for the* **shot05** *case.*

(normalized). A result of this inversion for the source coordinates using either of the objective functions is seen in Fig. 16 as a function of the number of frequencies in the inversion. The normalized-version results have unpredictable behavior, especially in the case of source depth, where the estimate does not converge at all. The unnormalized objective function seems preferable, and is standard for SAGA. This plot can easily be reconstructed using the **results.m** file. For further details, see [60, 61].

11.5.5 SNAPRD example. File: **tl_malta_rd**

This example relates to data taken during the Winter Sun experiment. We are inverting coherent transmission loss at four depths and four frequencies, over ranges from 200 m to 17 km; the data are given using the data format option **D**. We are matching both the shape and the offset of the 16 curves, option **X**. As the

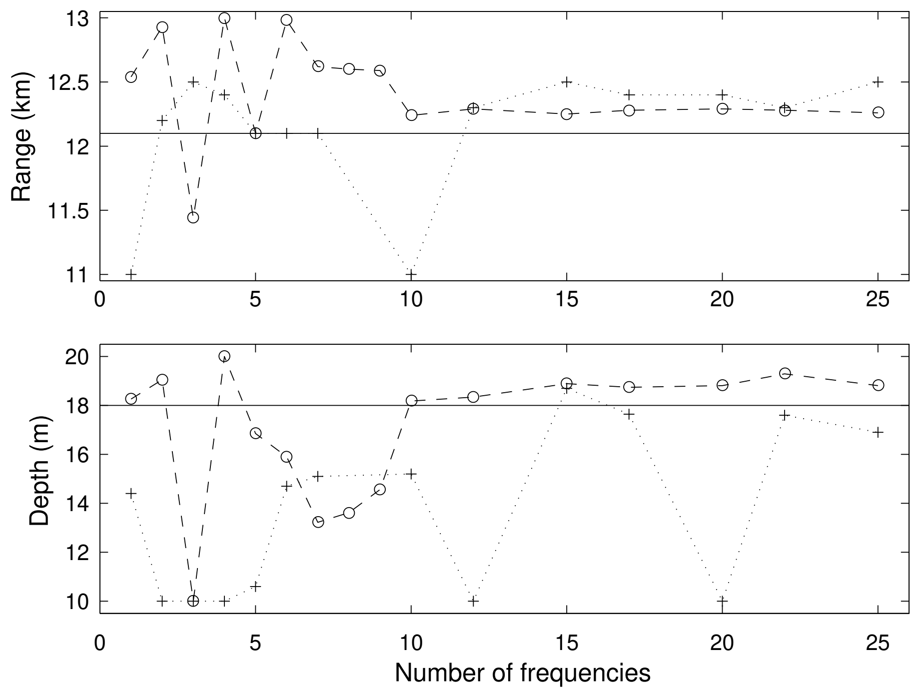

Figure 16 *Parameter estimates for source range (a) and source depth (b) for the* **shot05** *case. Solid lines indicate the baseline model values. Lines designated with crosses represent the estimates of the normalized objective function. Lines designated with circles indicate the estimates of the unnormalized objective function.*

final result of the inversion is presented as transmission loss curves plotted in dB, we are inverting the transmission loss measured in dB, option **l**, rather than using pressure magnitude. The bottom was slightly range dependent and therefore it was modeled with adiabatic modes with a new sector each 2 or 3 km. There is not enough information in the data to give an independent estimate of the bottom for each sector, therefore all bottoms were coupled together to find one representative bottom; this coupling of the parameters is done by using shape functions, option **E**.

Results of the inversion are shown in Fig. 15.

11.6 Forward model: PROSIM

PROSIM [5] is a broadband adiabatic normal mode program that is based on the ORCA program [6, 7]. In order to increase robustness and efficiency the search for eigenvalues is limited to the real wavenumber axis. Attenuation is accounted for using the same approximations as in SNAP. Shear and surface roughness is not yet included in the model. PROSIM is very fast for broadband normal mode calculations as it interpolates eigenvalues and mode functions as a function of frequency. However, if an error is detected in tracing the eigenvalues, a classical normal mode single frequency solution is used.

The main part of the input follows the description in the SNAP manual [1]. The frequencies that are used in PROSIM are specified by either:

```
Nfreq 0              !  the ''0'' is not read
freq(1) freq(2)...  freq(Nfreq)
```
or
```
-1 Δt                !  the Δt is only read for option F
Dfreq Fmin Fmax
```

For option **F** an unknown starting time has been introduced. This is due to the fact that we often do not know the absolute time, or that the frequency spacing is too large to obtain the full signal arrival time.

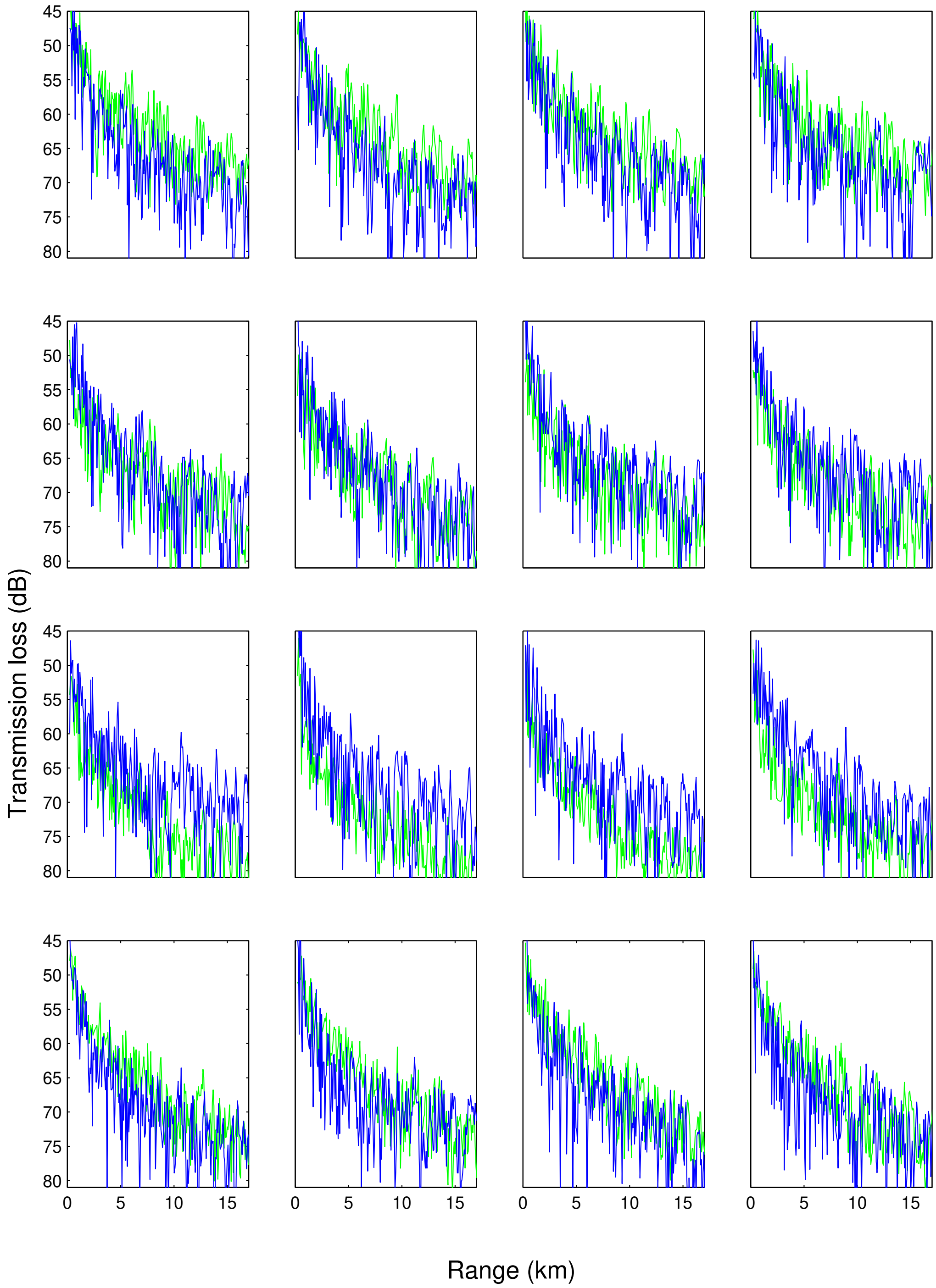

Figure 17 *Transmission loss for the* **tl_malta_rd** *case. The observed data (green line) and the modeled data (blue line) with the best found environment is plotted. The plot in each column represents frequencies 299, 399, 599, 998 Hz and each row from the top, hydrophone depths 121, 101, 81 and 61 m.*

11.6.1 Options in PROSIM

In the input file an option line is introduced for PROSIM:

p Positive gradient. Both sound speed and density in sediment and subbottom must increase as a function of depth. If, during optimization of an environment, negative gradients are found, the forward model will not be executed.

t Tilt of the vertical receiver array. It is measured as the horizontal deviation at the last receiver measured in meters. When this option is specified, it is the fourth parameter in the receiver-depth line of the input file.

T Two-way propagation. The transfer function from source to receiver is squared.

11.6.2 Pointers in PROSIM

Due to the increased dimensionality of the problem we need 3 pointers to specify which parameter we are inverting for. The two first pointers `parm` and `index` are nearly the same as in SNAP, while the third pointer `index2` points to the range sector.

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not used (e.g. source depth). The pointer `parm` can take the following values:

1 Water depth. The depth is changed by moving the last point in the water.
2 Sound speed in water; `index` refers to the actual point.
3 Sound speed in sediment; `index` refers to the actual point.
4 Attenuation. Depending on the value of `index` it refers to the following attenuation: `index` $= 1$(sediment), 2(bottom).
6 Sediment density.
8 Source depth.
9 Source-receiver range.
11 Shape-function coefficient; `index` points to the number.
12 Bottom P-speed.
14 Bottom density.
15 First receiver depth. The spacing is constant, thus it is a vertical translation.
16 Depth of speed point in water; `index` refers to the actual point.
17 Sediment thickness. The thickness is changed by moving the last point in the sediment.
18 Depth of speed point in sediment; `index` refers to the actual point.
19 Tilt. PROSIM option **t** must be specified.
20 Time delay ΔT.

11.6.3 PROSIM example. File: **waa**

This example is from the Matched Field Workshop, Vancouver, 1997 [62]. The options indicate that we are using a pressure vector, option **e**. The Bartlett power is added incoherently across frequencies, option **f**. We use data from 25 to 500 Hz with a spacing of 5 Hz, a total of 96 frequencies are used. For most other normal mode codes the CPU requirement for so many frequencies is so large that it is not practical to carry out an inversion. For the observed data and the replica with best estimated environment, the variation of power across the array is plotted, option **p**. We use GA with 20 populations, each with 1000 individuals; 20,000 forward models are used in total.

```
 waa.dat, 96 frequencies all phones.
f e p                   ! options
 1000 32 20            ! niter, q, npop
  0.8 0.5 0.05         ! px, pu, pm
p                      ! prosim option
 -1    0               ! number of frequencies
5 25  500              ! frequencies, Df, Fmin, Fmax
1                      ! number of sectors
 1 00  00              ! dum, phase speed (00 = all), dum

100      0    0  10. 0.   !depth, dum, dum, range, dum
   0.  1480.
  100  1460.
50     1.500 -0.23       ! sediment
   0.  1550
  50   1700
2.00 -0.23  1800.0       ! bottom
   0  0                  ! not used

20                       ! source depth
1 99.9999 100            ! rec depth
1                        ! number of ranges
5000.                    ! receiver range

 9                       ! nparm
 9  1  1 5000 5400 128   ! source range
 8  1  1   10   30 128   ! source depth
 1  1  1  100  120 128   ! water depth
 17 1  1   10   50 128   ! sediment thickness
 3  1  1 1500 1600 128   ! sound speed sed
 3  2  1 1550 1750 128   ! sound speed sed
 12 1  1 1600 1800 128   ! sound speed sed
 6  1  1 1.40 1.85 128   ! density in sediment
 14 1 1 1.600 2.00 128   ! density in half space
```

The *a posteriori* distributions for this case is shown in Fig. 18. When the distributions cluster around a certain value it is an indication that a parameter is well determined. The best obtained fit is shown in Fig. 19.

11.6.4 PROSIM example, file: **mf**

This example is based on the **waa** case from the Matched Field Workshop, Vancouver, 1997 [62, 63]. The input file is nearly the same as in the previous example. Only one phone at a depth of 50 m and a range of 5 km is used, but data from frequencies from 25–200Hz with a spacing of 1 Hz is used coherently. In the option line, option **F** is introduced in stead of option **f**.

The best obtained fit is plotted both in the frequency domain (Fig. 20) and in the time domain (Fig. 21). The transmission loss plot is produced directly from SAGA, whereas the time series first requires an FFT of the data. It should be noted that in the matched field approach used here, only the relative phase across all frequencies is determined. In order to plot the comparison of the time series the average phase difference between the observed and computed data was determined. The time series have been convolved with a Ricker wavelet with a center frequency of 80 Hz. Finally, due to the coarse frequency sampling the absolute arrival time cannot be determined.

11.7 Forward model: CPROSIM

Similarly to PROSIM the CPROSIM is based on the ORCA program [6, 7]. In order to also produce a correct field also at close ranges also the non-propagating eigenvalues are found in complex plane.

The input structure and the options are similar to PROSIM.

11.8 Forward model: POPP

The normal mode program POPP is used together with the reverberation model as described in Ref. [4]. The backscattering is estimated using Lambert's law with a power coefficient. The data should be expressed in dB and the matching is done in the dB domain. For an example of using this module, see Ref. [64] and Fig. 22.

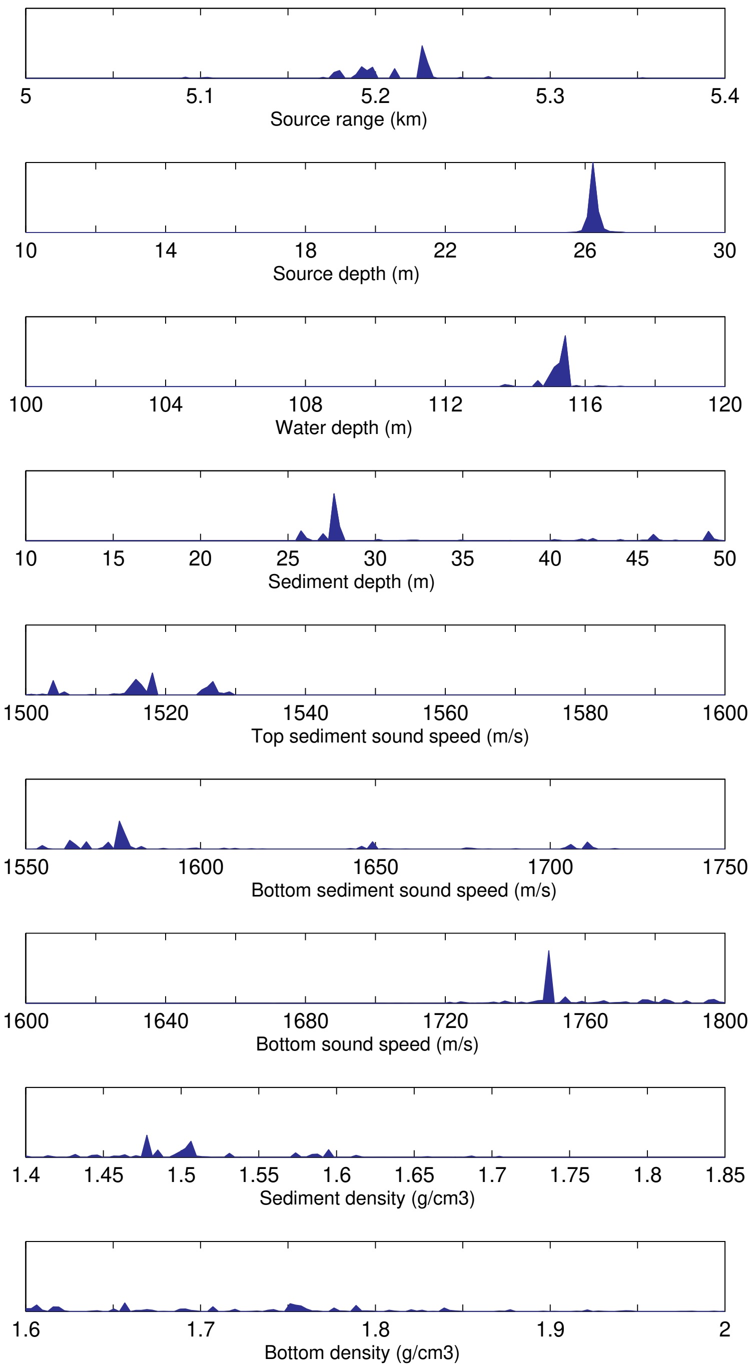

Figure 18 *The a posteriori distributions for the* **waa** *case.*

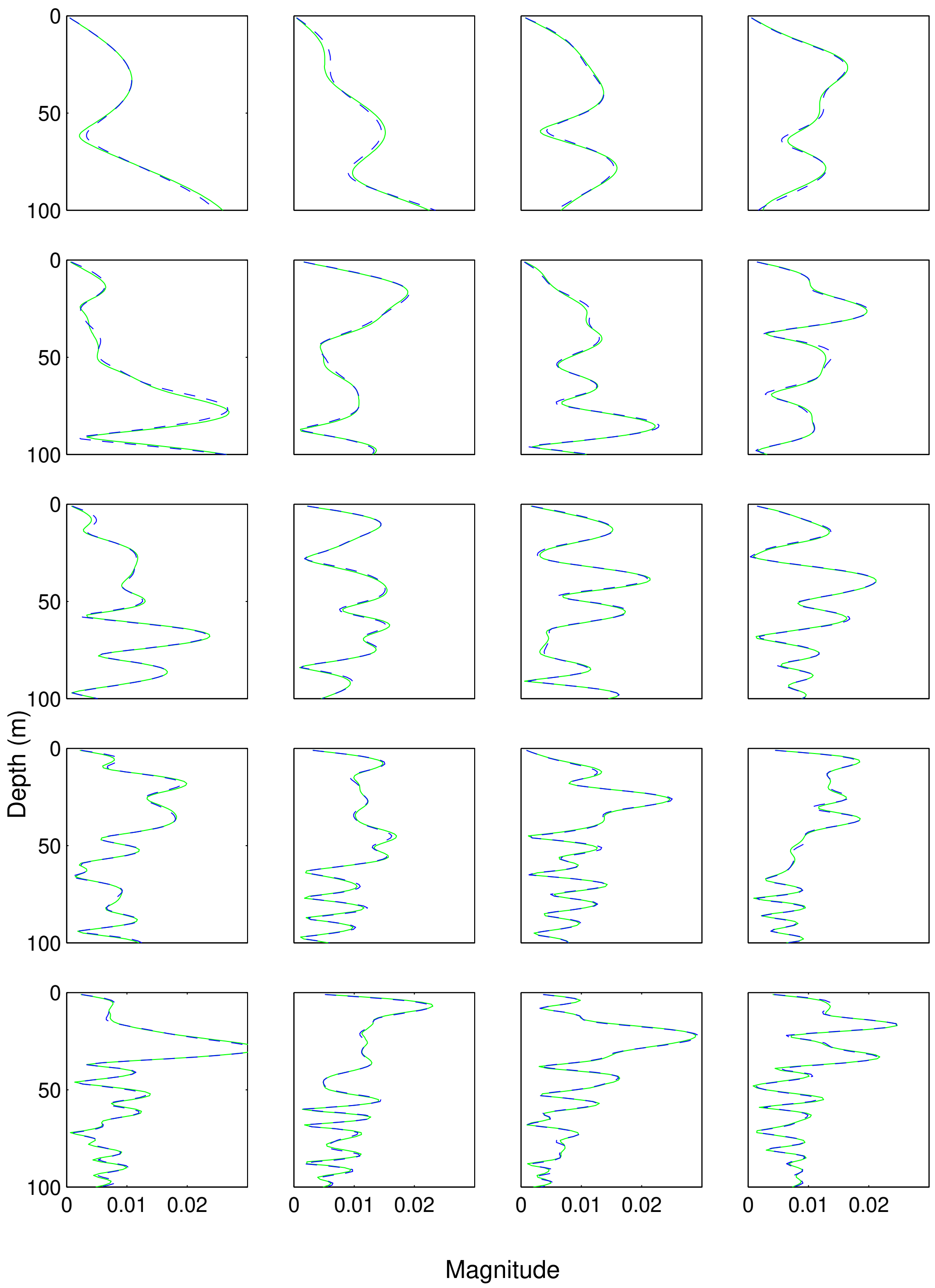

Figure 19 *Magnitude of the field for the* **waa** *case at every fifth of 96 frequencies. The observed data (green line), and the computed data (blue line) with the best set of parameters at every fifth frequency, i.e. from 25 Hz to 200 Hz in steps of 5 Hz.*

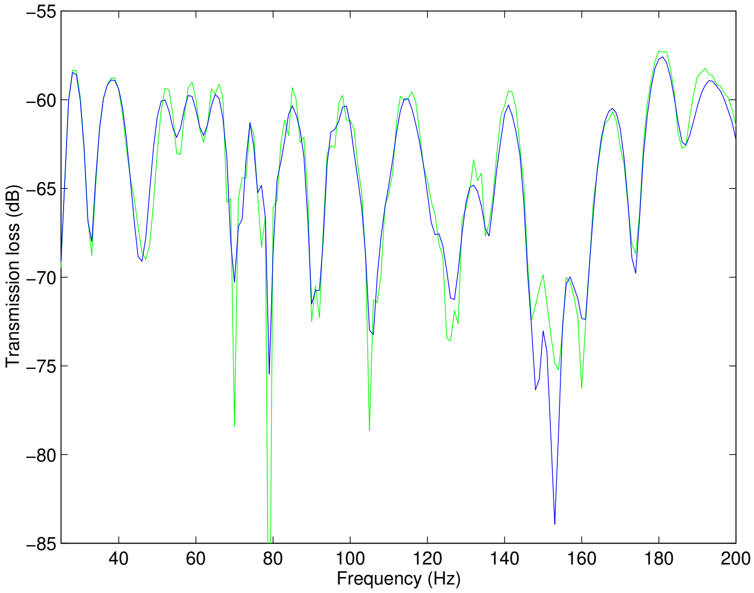

Figure 20 *The transmission loss for the* **mf** *case. The green line is the observed data and the blue line the modeled data with the best environment. The agreement is quite good.*

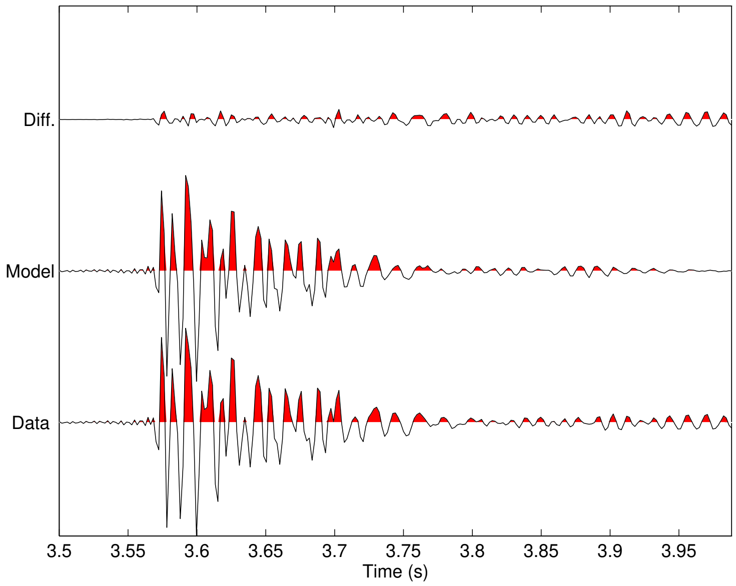

Figure 21 *The time series for the* **mf** *case. The observed data, the modeled data and the difference between the two time series are plotted.*

11.8.1 Pointers in POPP

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not important (e.g. source depth). The pointer `parm` can take the following values:

- **1** Water depth. The depth is changed by moving the last point in the water.
- **2** Sound speed in water; `index` refers to the actual point.
- **3** Sound speed in sediment; `index` refers to the actual layer.
- **4** Attenuation in sediment; `index` refers to the actual layer.
- **5** Thickness of each layer in sediment; `index` refers to the actual layer.
- **6** Density in sediment; `index` refers to the actual layer.
- **8** Source depth.
- **9** Source-receiver range
- **11** Shape-function coefficient; `index` points to the number.
- **15** Receiver depth.
- **16** Depth of speed point in water; `index` refers to the actual point.
- **17** Lambert scattering strength.
- **18** Power in Lambert's law.

11.8.2 POPP example. File: **revpopp**

This example is based on reverberation data from the North Elba site. There is not very much information in the data, thus we are mainly inverting for the Lambert scattering strength and the sound speed of the bottom half space. The options indicate that we are reading observed data using option **d**, matching the absolute level, option **X**, and we are plotting the observed data and the best fitting model, option **p**. We use GA with 1 population with 1000 individuals; 1000 forward models are used in total. The ambient noise level is determined from the data.

```
Elba reverberation data
d X p                        ! options

1  64  1                     ! niter, q, npop
0.8 0.5 0.05                 ! px, pu, pm

130.   1.0   0.              ! water depth, density, attenuation
    0.   1521.
   36.   1519.
   38.   1514.
   50.   1511.
```

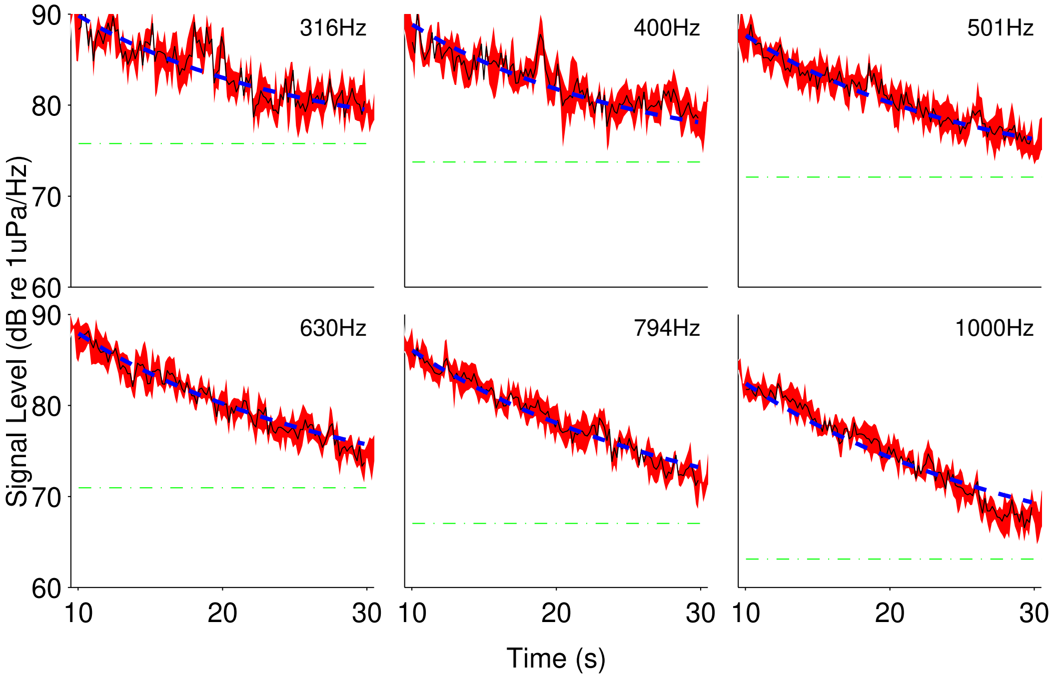

Figure 22 *Inversion of reverberation data from several shots (***File: popp_grad3***). The green dash-dotted line shows the estimated noise level in the data, the red area indicates the variation of the five used shots, the black thin full line is the average signal level of the shots. This is here computed as the first eigenvector of the co-variance matrix, where the covariance matrix is constructed as an ensemble average of several shots. The thick blue dashed line shows the synthetic data generated from the best estimated model.*

```
  130.   1508.
 -1.    -1.                ! no more water points
  0.   0.                  ! shear values not implemented
 -1. 1.8  1600.  0.09      ! sed thknss, dens, speed, att (-1= last lay)
  0.   0.                  ! roughness surface, bottom
  2   398   630.           ! Nfreq, freq
  1   60.                  ! Nsd sd
  1   60.                  ! Nrd rd
 -1   -1                   ! No. of comp. layers, modes (-1 = default)
2                          ! isb (1=surf, 2=bott)
  1                        ! tau0
198.8 197                       ! source level 1...Nfreq
-30   -30                        ! lambert coefficient 1...Nfreq
76    76                          ! ambient noise level  1...Nfreq

10.03067 30  0.1706666     ! tmin, tmax, tinc

 6                         ! nparm
 8  1  50   70    128      ! source depth
 3  1  1520 1620  128      ! vel sed 1
 1  1  120. 140.  128      ! water depth
 15 1   50. 70.  128       ! recvr depth
 17 1   -50 -20  128       ! lambert DMU for first frequency
 17 2   -50 -20  128       ! lambert DMU for second frequency
```

11.9 Forward model: GAMA

GAMA [10, 11] is developed by Evan Westwood and this version of GAMA has been ported into SAGA by Peter Nielsen.

GAMA is a broadband propagation model that uses ray theory to predict the acoustic field in a range-invariant, layered bottom ocean environments.

11.9.1 Options in GAMA

In the input file an option line is introduced for RAMGEO: There is currently no options for ramgeo.

- **p** Positive gradient. Both sound speed and density in
- **t** Tilt of a vertical array
- **T** Tilt of a horizontal array
- **h** Envelope of impulse response

11.9.2 Pointers in GAMA

Due to the increased dimensionality of the problem we need 3 pointers to specify which parameter we are inverting for. The third pointer `index2` points to the range sector.

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not used (e.g. source depth). The pointer `parm` can take the following values:

1 Water depth. The depth is changed by moving the last point in the water.
2 Sound speed in water; `index` refers to the actual point.
3 Sound speed in sediment; `index` refers to the actual point.
4 Attenuation. Depending on the value of `index` it refers to the following attenuation: `index` $= 1$(sediment), 2(bottom).
6 Sediment density.
8 Source depth.
9 Source-receiver range.
11 Shape-function coefficient; `index` points to the number.
12 Bottom P-speed.
13 Bottom S-speed. The shear in prosim should have a low value (about < 500 m/s) otherwise the perturbation correction is not valid.
14 Bottom density.
15 First receiver depth. The spacing is constant, thus it is a vertical translation.
16 Depth of speed point in water; `index` refers to the actual point.
17 Sediment thickness. The thickness is changed by moving the last point in the sediment.
18 Depth of speed point in sediment; `index` refers to the actual point.
19 Tilt for a vertical array. GAMA option **t** must be specified.
20 Time delay ΔT.
21 Tilt for a horizontal array. GAMA option **T** must be specified.
22 BLUB parameter 1.
23 BLUB parameter 2.
24 drtemp.

11.9.3 GAMA example. File: **map2k_phla2000067091700**

This example is from the MAPEX2K sea trial, The options indicate that we are using a pressure vector, option **e**. The Broadband matched filter is used as objective function option **F**. We use data from 240 Hz to 760 Hz in steps of 10 Hz. For the observed data and the replica with best estimated environment, the variation of power across the array is plotted, option **p**. For describing the environment we use shape functions option **E**. We use GA with 10 populations, each with 1000 individuals; 10,000 forward models are used in total.

```
MAPEX2K data on horizontal array! comments
W F E e p
1000 32 10
0.8 0.5 0.05
T                      ! Gama options
-1 0                   ! frequencies
10.000 240.000 760.000 ! 10 Hz bandfrequencies

1 1
1 0 0
130.3000  0.0  0.0 20.0  0.0  ! bathymetry
0.0000 1508.6530
5.2000 1508.6530
69.6000 1509.9090
101.7000 1512.000
130.3000 1512.3000
4.3000 1.5000 0.4000
0.0000 1559.0000
4.3000 1559.0000
1.8000 0.4000 1637.0000
0.0000 0.0000

55.0000                           ! source depth
50.0000 50.0000 1.0000 0.0000     ! rec depths
-128.0000
300.0000 554.0000

10                           ! nparm
11 1 1 1450 1700 128    ! upper velocity point (1st layer)
11 2 1    0  250 128    ! velocity inc (2nd layer )
11 3 1    0    1 128   ! attenuation
11 4 1 1.05  2.5 128   ! density
17 1 1  0.1   20 128   ! thickness sediment layer
9  1 1  305  307 128   ! source range
8  1 1   55   57 128   ! source depth
1  1 1  129  131 128   ! water depth
15 1 1   54   57 128  ! receiver depth
```

```
21 1 1   -2    2 128  ! horizontal tilt
```

11.10 Forward model: ORCA

ORCA90 [6, 7] is developed by Evan Westwood and is here used in a subroutine version supplied by Dag Tollefsen [8] based on ORCA v3.0 [**?**]. Note that this version can only run in range-independent two-dimensional waveguide with a layered fluid-solid seabed, a single multi-frequency source and a linear vertical or horizontal acoustic array [8]. In the real axis version it is considerably faster than PROSIM or SNAP .

11.10.1 Options in ORCA

In the input file an option line is introduced for ORCA: There is currently no options for ramgeo.

- **H** Horizontal array
- **V** Vertical array
- **t** Tilt (straight-line) of Vertical Array
- **C** Using ORCA complex root finder
- **p** Allow only positive velocity gradients

11.10.2 Pointers in ORCA

Due to the increased dimensionality of the problem we need 3 pointers to specify which parameter we are inverting for. The third pointer `index2` points to the range sector.

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` and `index2` is not used (e.g. source depth). The pointer `parm` can take the following values:

1 Water depth. The depth is changed by moving the last point in the water.

2 Sound speed in sediment; `index` refers to the actual layer. `index2` points to top (1) or bottom (2) of layer, `index2`= 0 points to both top and bottom, i.e a constant sound speed layer.

3 shear Sound speed in sediment; `index` refers to the actual layer. `index2` points to top (1) or bottom (2) of layer, `index2`= 0 points to both top and bottom, i.e a constant sound speed layer.
4 Attenuation in sediment; `index` refers to the actual layer. `index2` points to top (1) or bottom (2) of layer, `index2`= 0 points to both top and bottom, i.e a constant sound speed layer.

5 Attenuation in sediment; `index` refers to the actual layer. `index2` points to top (1) or bottom (2) of layer, `index2`= 0 points to both top and bottom, i.e a constant sound speed layer.

6 Attenuation in sediment; `index` refers to the actual layer. `index2` points to top (1) or bottom (2) of layer, `index2`= 0 points to both top and bottom, i.e a constant sound speed layer.

7 Layer Thickness; `index` refers to the actual layer.
8 Source depth.
9 Source-receiver range
11 Shape-function coefficient; `index` points to the number.
15 Receiver depth.
19 Tilt. ORCA option **t** must be specified.
20 Ocean sound speed. `index` points to the sound speed point.

11.10.3 ORCA example. File: **sdc**

This example is from the Matched Field Workshop, Vancouver, 1997 [62]. The options indicate that we are using a covariance matrix, option **c** normalized with the trace option **b**. We use data from at only 100 Hz. We are using enumerative integration to look at the posterior distributions option **S2**. After 2000 forward models the convergence is tested and an error between the marginal distributions less than 0.1 terminates the run.

```
 sdc.dat test case  orca
S2 c  b   z                ! options
2000  32  10               ! niter, q ,npop
0.8 0.5 0.05  20           ! px,pu,pm
0.00833 0.1 0.1 2          ! nu   e_stop e_rot k_grow0.033
V                          ! orca90 options
1                          ! number of frequencies
100                        ! frequencies

 2                         ! number of SSP points
    0.0000 1480.00
  100      1460.00
```

```
 1                        ! number of bottom layers
30.6873 1530.44,1604.15 0,0 1.5008,1.5008 -0.23,-0.23 0,0
1689.0 0.0 1.70064 -0.23 0.0

    20                    ! source depth
  5 99.9999 20            ! rec depth
 1                        ! number of ranges
 1                        ! receiver range (km)

4                         ! nparm
7  1  1  10   50    30   ! sediment thickness
2  1  1  1500 1600   30  ! sound speed sed
2  1  2  1550 1750   30  ! sound speed sed
2  2  0 1600  1800  30   ! sound speed sed
6  1  1 1.400 1.850 20   !density in sediment
14 1 1 1.600 2.000 128   !density in half space
```

11.11 Forward model: RAMGEO

RAMGEO [12, 13] is developed by Mike Collins. The "GEO" version of RAM handles multiple sediment layers that parallel the bathymetry. This is efficient for geoacoustic inversions.

For RAMGEO the CPU time depends on the discretization in depth (dz), range (dr) and Padé order (ϕ) and well as the maximum depth and range point. ($CPUtime \propto (\frac{1}{dz})^2(\frac{1}{dx\phi})^{0.8}$). In order to get a reasonable low CPU time it is important to have good values of these parameters.

It is necessary to run convergence test to assure that the solution has converged. With SAGA this can easily be done running SAGA with just one forward model and then by copying the obs-file to the in-file. Then change the discretization and observe the fitness between the two models. This fitness-value should be below the noise floor of the data.

11.11.1 Options in RAMGEO

In the input file an option line is introduced for RAMGEO: There is currently no options for ramgeo.

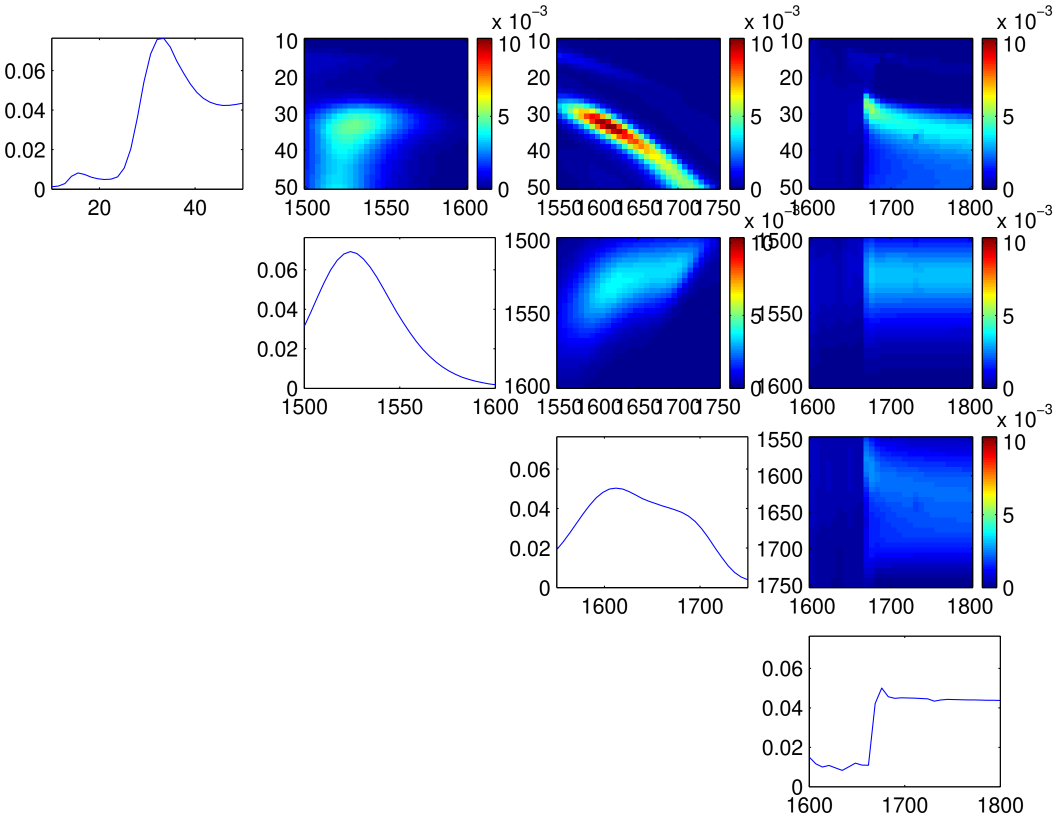

Figure 23 *A posteriori distribution of the sediment thickness, sound speed (top and bottom of sediment layer and bottom layer) from the sdc case in [62]. The the diagonal shows the marginal distribution. and the off-diagonal the 2D marginal distributions.*

11.11.2 Pointers in RAMGEO

Due to the increased dimensionality of the problem we need 3 pointers to specify which parameter we are inverting for. The third pointer `index2` points to the range sector.

The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not used (e.g. source depth). The pointer `parm` can take the following values:

1 ocean sound speed , `index` points to depth and index2 points to range.

2 bottom sound speed,`index` points to depth and index2 points to range.

3 bottom density, `index` points to depth and `index2` points to range.

4 bottom attenuation, `index` points to depth and `index2` points to range.

5 ocean sound speed *depth*, `index` points to depth and index2 points to range.

6 bottom sound speed *depth*, `index` points to depth and index2 points to range.

7 bottom density *depth*, `index` points to depth and `index2` points to range.

10 bottom attenuation *depth*, `index` points to depth and `index2` points to range.

8 source depth (`zs`).

9 source range (`rmax`).

11 shape functions index points to shape function.

12 bathymetry `index` points to point.

13 bathymetry range point, `index` points to point . point.

11.11.3 RAMGEO example. File: **tc1v1**

This example is from the Inversion Techniques Workshop, 2001 and described further in Ref [74]. The options indicate that we are using a pressure vector, option **e**. The Bartlett power is added incoherently across frequencies, option **f**. We use data from only 300 Hz. For the observed data and the replica with best estimated environment, the variation of power across the array is plotted, option **p**. For describing the environment we use shape functions option **E**. We use GA with 10 populations, each with 1000 individuals; 10,000 forward models are used in total.

```
Inversion Workshop, Test Case 1 RAM Input File
E e f1 p                ! options
1000  32  10            ! niter, q ,npop
0.8 0.5 0.05            ! px,pu,pm
                        ! ram options

1                       ! #freq,
 300                    ! freq
 20                     ! ZS
500 500.0 0
 5                      ! RMAX DR NDR
300.0  0.1  10 ! ZMAX DZ NDZ
20 80                   ! rdmin,rdmax
1488.6 5 1 0.0          ! CO NPADE NS RS

     0.0        90.0
  5000.0       150.0
-1  -1
   0.     1495.0
 160.     1488.6
-1   -1
   .00    1535.
  3.00    1541.4
  3.00    1541.4
 10.00    1556.4
 10.00    1556.4
 35.00    1610.
 35.10    1800.
-1   -1
  .000          1.550
  34.900        1.590
  35.100        1.950
-1   -1
   .0       0.123
 35.0       0.032
 35.0       0.036
100.0       0.036
150.0      10
300        100
   -1   -1
   -1   -1

13                           ! nparm
11  1  1  1500 1600 128      ! upper velocity point (1st layer)
11  2  1  1    150  128      ! velocity inc (2nd layer )
11  3  1  1    150  128      ! velocity inc (3rd layer )
11  4  1  10   400  128      ! velocity inc (half space layer )
11  5  1  2    10   128      ! thickness first  layer
11  6  1  2    20   128      ! thickness second layer
```

```
11  7  1  2    30   128      ! thickness third  layer
11  8  1  1.2  1.8  128      ! upper density
3   3  1  1.6  2.4  128      ! lower density
11  9  1  0.01 0.5  128      ! upper attenuation
11  10 1  0.01 0.2  128      ! lower attenuation
12  1  1   89  91   128      ! Bathymetry  point (r=0)
12  2  1  149  151  128      ! Bathymetry  point (r=5km)
```

11.12 Forward model: TPEM

The Terrain Parabolic Equation Model (TPEM) calculates the electromagnetic field in height and range for electromagnetic propagation in the troposphere. It allows for range-dependent refractivity environments and variable terrain. TPEM is based on a source code originally developed by Fred Tappert from the University of Miami, for propagation over a smooth surface. It is a pure PE model based on the split-step Fourier method and is described in [14, 15] (Email: barrios@nosc.mil). A good general description of electromagnetic field propagation in the troposphere is given in [65]. The version used in SAGA is based on TPEM 1.0 originally developed for a PC.

For a parabolic equation method it is required that the solution converge for smaller step sizes in range and depth. The size of these steps depend on the particular environment and the terrain profile. Based on a few environments we have empirically found the following step sizes to be adequate: About $\lambda/3$ in depth, and in range either 10λ for range-dependent or 200λ for range-independent environments. For range-dependent environments smaller steps are required because the field generally varies more with range. These limits are implemented in the broadband version of TPEM and are significantly larger that the ones used in [15].

In SAGA the range steps are increased by a factor 5, i.e. to 50λ for range-dependent and 1000λ for range-independent environments. The step size in meters are written out during execution of the code. A doubling in either step size causes roughly a halving in CPU time and, therefore, a significant CPU-time saving can be obtained by changing these default values.

The default step sizes can be changed by specifying a TPEM option **u** as indicated below.

The maximum height is specified in the input file (either as antenna height or maximum refractivity profile height). This height combined with Δz will determine the number of points needed in the Fourier transform. Therefore, in order to reduce CPU-time these heights should be as small as possible. The program is also much

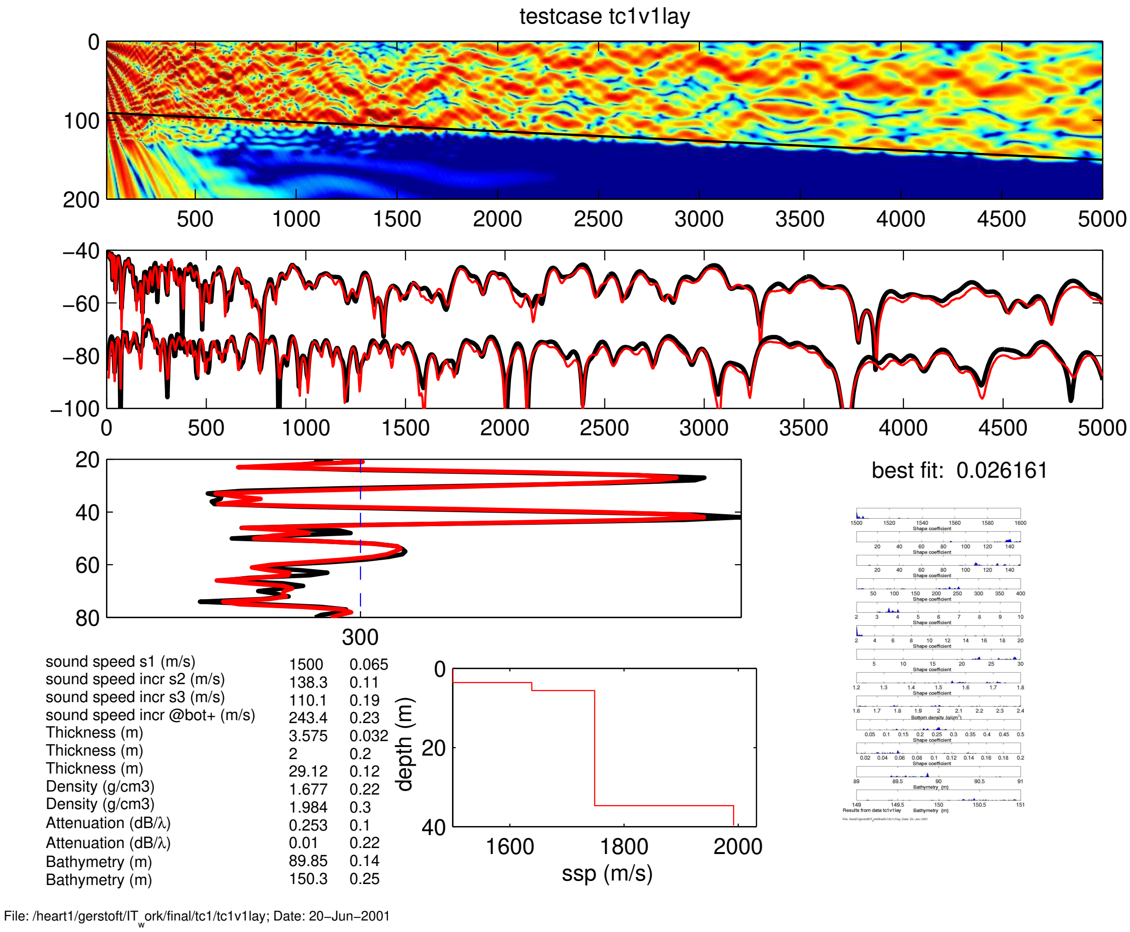

Figure 24 *Inversion of vertical array data from the 2001 Geoacoustic inversion workshop (***File: tc1v1lay***). A contour plot for the best matching field (top). Comparison of observed TL and TL from the best model at 200Hz (top middle). The match of the data and modeled field on the vertical array for frequency used in the inversion. (bottom left) The obtained parameters and their normalized standard deviation (bottom left). The obtained velocity profile (bottom middle). The a posteriori distributions (bottom right). A contour plot for the best matching field (top). Comparison of observed TL and TL from the best model at 200Hz (top middle).*

faster if running in a range-independent mode, which is specified by using "0" terrain points. This is partly because larger range steps can then be taken.

The magnitude of the field is best computed by using the SAGA option **G**. If the magnitude of the field is required in dB, then use option **l**.

11.12.1 Options in TPEM

u (itpem_opt(1)=1)user defined step sizes for sampling of the field. An extra line should then be inserted below the "center frequency line".

`dr dz ln`

`dr dz` is specified in meters and `ln` is the FFT-power for the Fourier transform in height. The maximum height of the computational domain is $\mathtt{dz}\cdot 2^{\mathtt{ln}}$.

E (itpem_opt(2)=1) Reading shifted EOF functions for the refractivity profile. The next lines should then contain

`Neof` The number of EOFs used.
`Base-height` $\mathtt{Coef}_1$ ... $\mathtt{Coef}_{Neof}$

m (itpem_opt(3)=1) Magnitude of field

c (itpem_opt(4)=1) Clutter return is modeled. The formula is

$$p_c(r, \mathbf{m}) = -40 \log f(r, \mathbf{m}) + 10 \log(r) + c(\mathbf{m}) + \sigma(r) \tag{36}$$

where c is an unknown range-independent constant. f is the 1-way modeled propagation loss as modeled by 2-D PE TPEM. $\sigma(r)$ represents the variation in the clutter cross section. Usually we assume $\sigma(r)$ to be constant,but range dependent is also possible (see Sect XX). The assumption constant $\sigma(r)$ breaks down when there is rain or interfering ships. These must first be removed from the clutter map.

Further, the coefficient $c(\mathbf{m})$ is adjusted so that the mean of the modeled clutter $p_c(r, \mathbf{m})$ is the same as the observed clutter. The clutter points with negative $p_c(r, \mathbf{m})$ is truncated.

The clutter return, $\sigma(r)$ is modeled as a using a linear variation between the value of the clutter cross section σ_i, $i = 1 \cdots N_{\text{clut}}$ at control points at range r_i, where N_{clut} is the number of clutter points.

In the input file the following parameter must be specified before the line with "refractivity profile points"

```
znoise
number-of-clutter-points
Xclut-1 Xclut-2 Xclut-3 ...
```

```
Cclut-1 Cclut-2 Cclut-3 ...
```

p (itpem_opt(5)=1) A parametric profile. Instead of specifying the M-profile for each altitude a set of parameters is used to describe the profile. the input format is (replacing the M-profile section
`base-height thickness M-offset M-deficit max-height`
`slope1 slope2 delta`

For example,

```
90 20 330 50 400 :  base-height, thick, offset deficit, zmax
0.13 0.118 0 0 :  slope mix, slope top, delta
```

The lower line is described as the coefficients for the M-profile.

a (itpem_opt(6)=1) The range dependence in base height is described by use of a set of polynomials. These polynomials is described in the `markov.in` file. We have determined these coefficients as eigenvalues of a Markov process. The coefficients to these polynomials is specified after the refractivity profiles with the following format:

```
3 :  polynomial for base height
100 -50 100 0 0 0 0 :  factors for base height shape function.
```

d (itpem_opt(7)=1) “ Multiple beam inversion”

r (itpem_opt(8)=1) “ Maximum M-deficit inversion”

e (itpem_opt(9)=1) “ Meteorologic constrain inversion”

11.12.2 Pointers in TPEM

TPEM uses three pointers to specify a variable. The pointer `parm` is used to map between the optimization variable and the environmental parameter to be optimized. The second parameter, `index`, points to the layer; for some parameters `index` is not important (e.g. source height). The pointer `parm` can take the following values:

1 (`rf.refmsl`) Refractivity points in M-units; `index` indicates at which height and `index2` at which range.
2 (`rf.hmsl`) Height corresponding to a refractivity point; `index` indicates at which height and `index2` at which range.
3 (`tr.terx`) Terrain x-coordinates.
4 (`tr.tery`) Terrain y-coordinates.
5 Source height.

6 (`sv.antht`) Antenna height.
9 Source-receiver range.
11 Shape-function coefficient; `index` points to the number.
12 base-height `par2lay` points to range.
13 thickness, `par2lay` points to range.
14 offset, `par2lay` points to range.
15 m-deficit, `par2lay` points to range.
16 noise-floor for TPEM option option **c**.
17 coefficient(type, range) for TPEM option option **p** for describing the tri-linear profile`par3` points to range.
18 Clutter cross section ccs(range) for TPEM option option **c**, `par2lay` points to range.
19 factor(`par2lay`) for TPEM option option **c**.

11.12.3 TPEM example. File: **tpem_ex**

This example has been used in paper [48]. First the data from the initial model is written to the *.obs file, option **W**. This file can then be copied to the *.in file and be used in subsequent inversions. The data format is the complex-valued field on a vertical array, option **e**. The objective function is incoherent Bartlett, option **f**. This type of environment is efficiently described using a tri-linear profile, see Fig. 25, where the refractivity profile is described in terms of M-deficit, base height and thickness. This type of profile is efficiently described using shape functions and read from the *.eof file, option **E**. When running POST, a plot of the observed data and the synthetic data for the best environment is produced, option **E**. We are optimizing over the three environmental parameters: M-deficit, base height and thickness.

```
TPEM optimizing for the tri-linear profiles
 W f e  p  E          ! options

  2000  64  20 ! niter, q, npop
  0.8 0.5 0.05 ! px, pu, pm

                 : Tpem-options
1000.            : Frequency in MHz (100 to 20000)
1                : Field type ( 1:complex field, 0:magnitude )
50.             : Transmitter height in meters
0               : Ant pattern(0=Omni, Gaussian, Sin(x)/x, Csc-Sq, Ht-finder)
30              : Beamwidth in degrees (full 3 dB to 3 dB width)
0               : Elevation angle in degrees
100            : Maximum receiver height in meters
90000.           : Maximum receiver range in meters
```

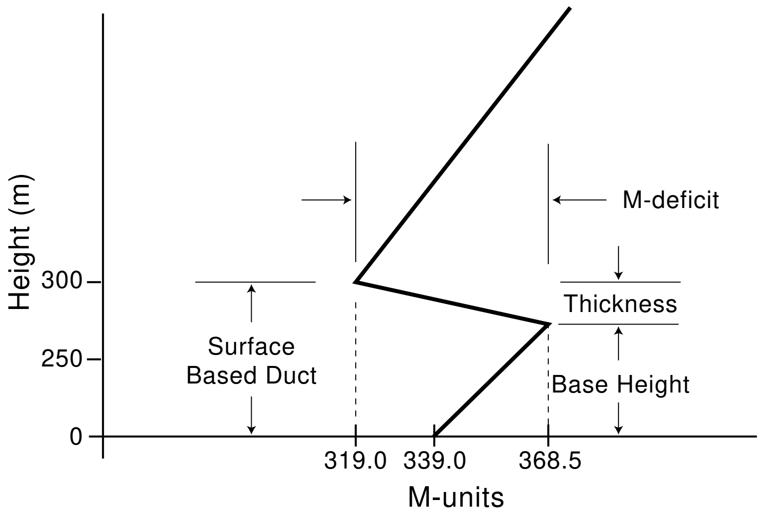

Figure 25 *Modified vertical refractivity profile for a tri-linear profile.*

```
1  50          : Number of range, height points to output
1   4           ! refractivity profile points (Range, Height)

0                 !First profile is at range zero
0.         339.         ! Refractivity
250.       368.5
300.       319.
400.       330.8

 0               ! terrain points => range-independent env.

 3  ! nparm
 11 2 1   1    100   128 ! refraction deficit
 11 3 1   1    400   128 ! base height
 11 4 1   1    100   128 ! thickness
```

The shape function file contains the three basic parameters used to describe the atmospheric profile in Fig. 25, see Ref. [48].

```
! shape function file for the TPEM
5  7  1       ! No. of shape functions, No. of points, No. of blocks
1 1  1        ! refractivity 1 (bottom)
1 2  1        ! refractivity 2
1 3  1        ! refractivity 3
1 4  1        ! refractivity 4 (top)
2 2  1        ! height
2 3  1        ! height
2 4  1        ! height (top)
```

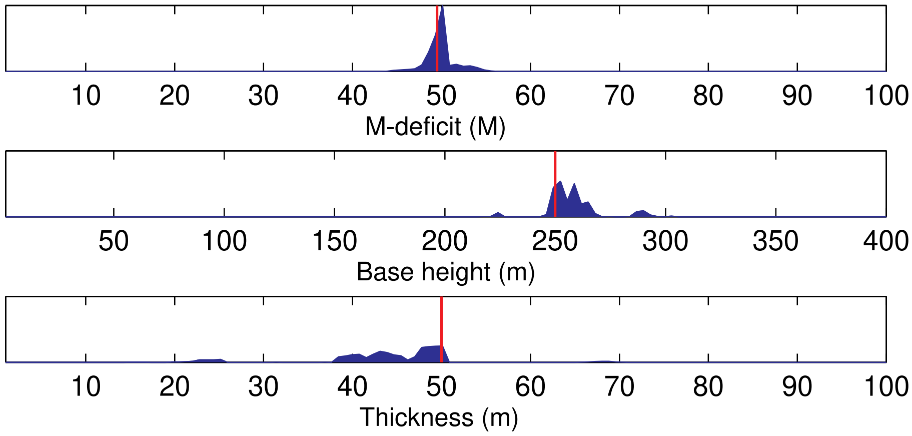

Figure 26 *The a posteriori distributions for the* **tpem_ex** *case. The red line indicates the true value.*

```
! Block 1: coupling of all parameters using the tri-linear profiles
! c(1) Deficit base thick top
5 7
   1      0     0      0    0        ! first refractivity point
   1      0   0.118    0    0        ! second
   1     -1   0.118    0    0        ! third
   1     -1     0   -0.118 .118      ! fourth
   0      0     1      0    0        ! second height
   0      0     1      1    0        ! third
   0      0     0      0    1        ! fourth
! Now comes the starting values
  339   49.5   250    50  1000        ! starting values
```

The *a posteriori* distributions for this inversion are shown in Fig. 26. When the distributions cluster around a certain value it is an indication that a parameter is well determined. The best obtained fit is shown in Fig. 27. Since there is no noise in the observed data the match is quite good.

It is also of interest to plot the ambiguity surface between two parameters. This will illustrate the relative importance of the two parameters. The contour plot is produced by introducing option **C** in the option line. When running SAGA it will then produce a contour plot of the objective function between the first two specified optimization parameters. In order to reduce the number of forward modeling runs, it is advisable to reduce the discretization for each of the two parameters to about 40. An example of an ambiguity surface is given in Fig. 28. The ambiguity surface is shown using either complex-valued received signal or just using the magnitude. The magnitude is produced by introducing option **G** in the option-line.

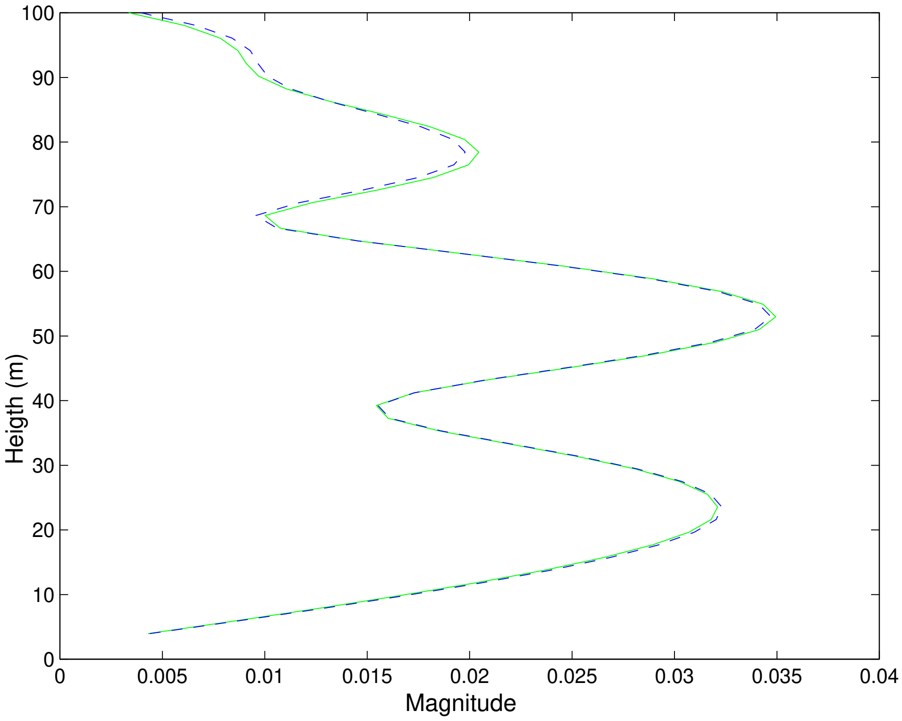

Figure 27 *The fit for the* **tpem_ex** *case. The magnitude of the observed data (green line), and the magnitude of the synthetic data (blue line) with the best set of parameters.*

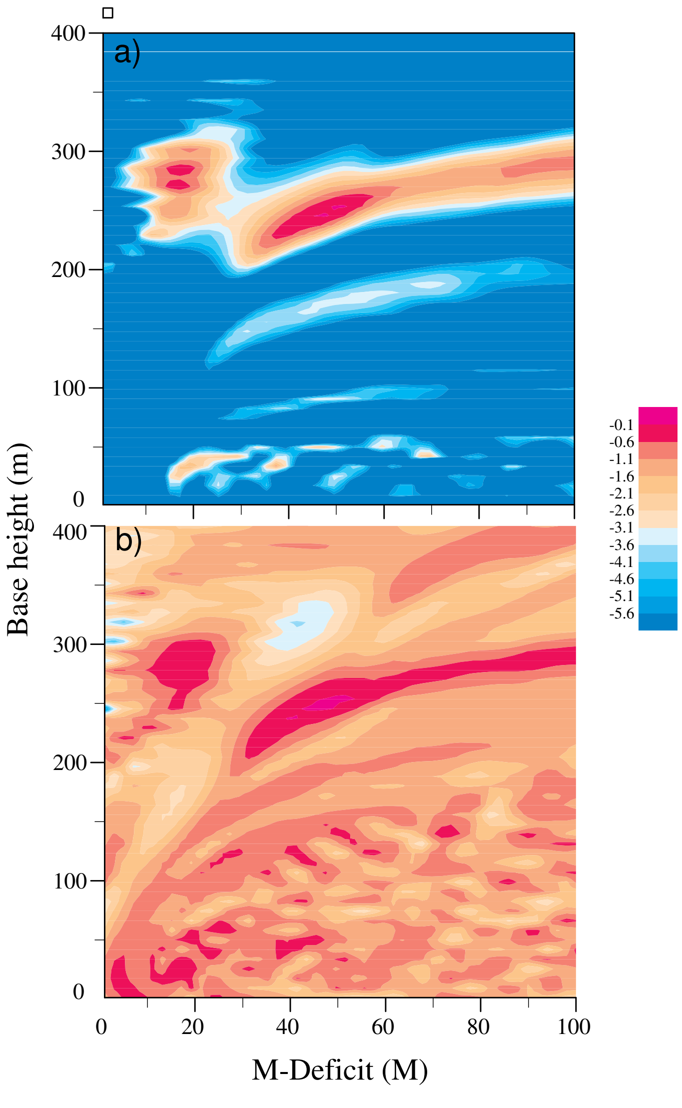

Figure 28 *Contour plot of the ambiguity function between base height and M-deficit for the* **tpem_ex** *case. Phase (a) or magnitude (b) of the received signal are used in the processing.*

12
3D array localization using SNAP

The purpose of this section is to describe how SAGA can be combined with SNAP to provide an ability to determine the shape of a receiver array in 3D. This ability can be combined with determination of environmental parameters. Snap only works in a plane geometry. The out of plane array is then projected into this plane. The vertical plane is defined by the source and the first receiver.

12.1 Geometry

Two coordinate systems are used for describing the geometry, see Fig. 29. As is usual the propagation code uses a global coordinate system with the z-axis pointing downwards and $z = 0$ at the surface, and the range $r = 0$ at the source. In order to describe the receiver array, a local right handed coordinate system is defined with origin at the first array element and the x-axis horizontal and pointing away from the source, the z-axis pointing vertically downwards. A positive rotation around each axis is defined as being clockwise when looking from the origin along the positive axis.

First the arrays deviation from a straight line is computed in the xz-plane with the axis of the array located along the z-axis and the deviation in the x-axis. Here the array is defined in terms of a parabola. Then this shape is rotated in 3D using the Eulerian angles[66]; first around the z-axis, then the y-axis and finally the z-axis again.

12.1.1 Parabola

The parabola is specified in terms of the bow a at the midpoint of the array and the length of the straight array L_s. It is specified along the z-axis and the deflection is given along the x-axis, as follows:

$$x = \frac{4a}{L_\mathrm{p}^2}(L_\mathrm{p} - z_\mathrm{p})z_\mathrm{p} \tag{37}$$

Here L_p is the length along the z-axis and z_p is the local z-coordinate.

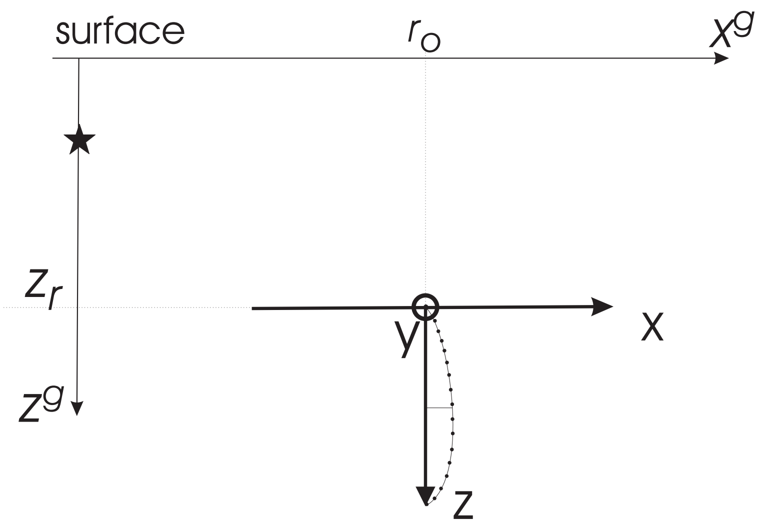

Figure 29 *The used coordinate systems. The local coordinate system has origin in the first receiver and is a simple translation of the global coordinate system. The y-axis is pointing out of the plane.*

The arc length L_s of the parabola is (this is also the length of the straight array)

$$\begin{aligned} L_\mathrm{s} &= \int_{z=0}^{L_\mathrm{p}} \sqrt{1+\left(\frac{\partial x}{\partial z}\right)^2} \mathrm{d}z = \frac{L_\mathrm{p}}{2}\left[\sqrt{1+b^2}+\frac{1}{2b}\ln\frac{\sqrt{1+b^2}+b}{\sqrt{1+b^2}-b}\right] \\ &\approx \frac{L_\mathrm{p}}{2}\left[(1+b^2/2)+\frac{1}{2b}(2b-\frac{b^3}{3})\right] \approx L_\mathrm{p}+\frac{8}{3}\frac{a^2}{L_\mathrm{p}} \end{aligned} \tag{38}$$

where $b = \frac{4|a|}{L_\mathrm{p}}$. To second order in a the length of the array along the z-axis is given by

$$L_\mathrm{p} = L_\mathrm{s}(1-\frac{8}{3}a^2/L_\mathrm{s}), \tag{39}$$

The original z-coordinates for a straight array z_s is then modified due to the bow of the parabola.

$$z_\mathrm{p} = z_\mathrm{s}L_\mathrm{p}/L_\mathrm{s} \tag{40}$$

For a given array, the element positions are known for the undeflected shape, z_s. Using the approximation above, Eq. (40) the z-coordinates of the element positions for a bowed array are calculated. The local x-coordinate is then calculated from Eq. (37).

12.1.2 Rotation

For a given shape of the array the orientation in 3D is then determined from rotations of the array. The array has its axis defined as the z-axis and the shape defined in terms of (x, y, z) coordinates. When using the parabola as defined above $y = 0$. This is done by the three rotations as defined by the three Euler angles[66]:
Rotation-1 the array is rotated θ_1 around z-axis,
Rotation-2 the array is rotated θ_2 around y-axis, and
Rotation-3 the array is rotated θ_3 around z-axis.
The rotations must be performed in this order. Performing the three rotations we obtain the following expression for the rotated array $(x'\ y'\ z')^{\mathrm{T}}$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \overbrace{\begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{rotation-3}} \overbrace{\begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix}}^{\text{rotation-2}} \overbrace{\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\text{rotation-1}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{41}$$

Examples of the rotation is shown in Figs. 30 and 31. In Fig. 30 the rotation $(\theta_1, \theta_2, \theta_3) = (90^o, 90^o, 45^o)$ gives a horizontal array pointing with a 45^o angle to the source-receiver line. In Fig. 30 31

A horizontal array is obtained by rotating $(\theta_1, \theta_2, \theta_3) = (90^o, 90^o, -)$. A vertical array is obtained by rotating $(\theta_1, \theta_2, \theta_3) = (-, 0^o, -)$ or $(-, 180^o, -)$. A positive bow a points towards the source when rotation3 is $0 - 180^o$. The bow and rotation3 shows negative symmetry.

The total horizontal range from the source to each receiver element is:

$$r = \sqrt{(r_0 + x')^2 + y'^2} \tag{42}$$

where r_0 is the range from the first array element. The global z coordinate used in the propagation code is found by adding the first receiver depth z_r to the local z-coordinate.

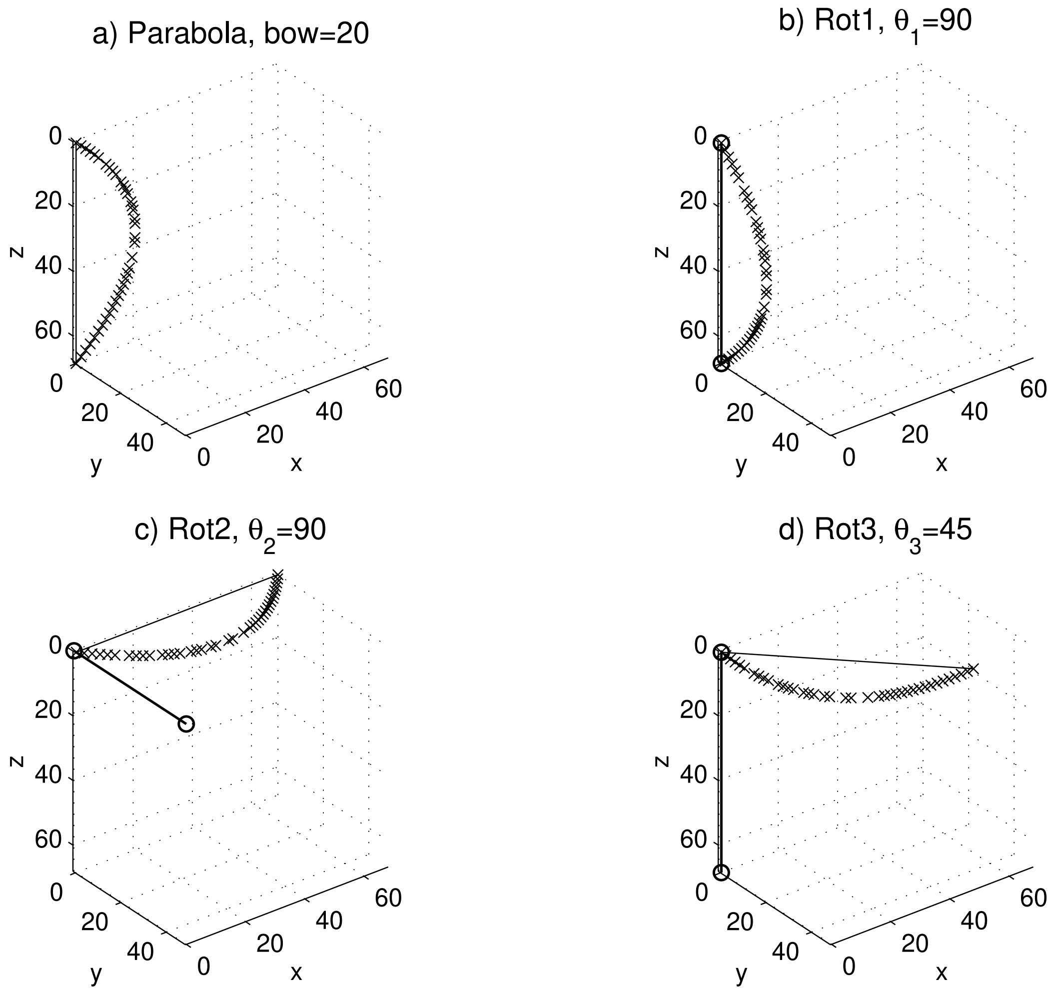

Figure 30 *A horizontal array with a 45^o angle to the source-receiver line. The rotation are* $(\theta_1, \theta_2, \theta_3) = (90^o, 90^o, 45^o)$. *a) The parabola is calculated in the xz-plane, b) The parabola then is rotated* θ_1 *around the z-axis, c) The parabola then is rotated* θ_2 *around the y-axis, d) The parabola then is rotated* θ_3 *around the z-axis.* [**hor45**]

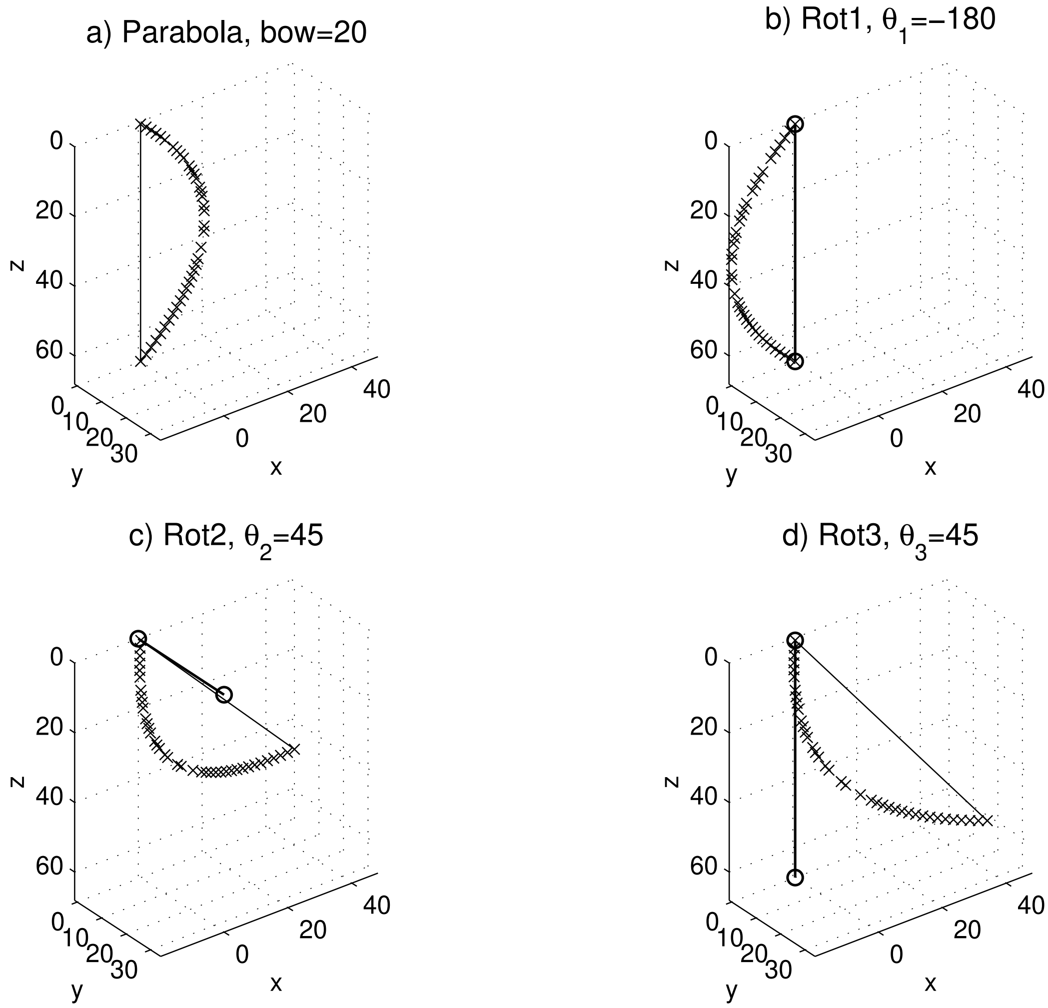

Figure 31 *A tilted array. The rotation are* $(\theta_1, \theta_2, \theta_3) = (180^o, 45^o, 45^o)$. *a) The parabola is calculated in the xz-plane, b) The parabola then is rotated* θ_1 *around the z-axis, c) The parabola then is rotated* θ_2 *around the y-axis, d) The parabola then is rotated* θ_3 *around the z-axis.* [**tilted**]

12.2 *Visualization of the rotations*

It can be quite difficult to understand the rotations in 3D. In order to visualize these rotations it is useful to try the matlab program `rotation3d.m` in the matlab directory. It was used to create the Figures 30 and 31.

12.3 *Additional Options in SNAP*

All of the options below define that out of plane geometry is used. The array can be defined in 4 ways:

- **o** Equidistant receivers; the deflection modeled as a parabola. The array is rotated.
- **x** The local z coordinates of each receiver are read from input file; the deflection modeled as a parabola. The array is rotated.
- **x2** The local z and x coordinates of each receiver are read from input file. The array is rotated.
- **x3** The (x, y, z) coordinates of each receiver are read from input file (global z and local x and y); No rotation is carried out.

The array shape is specified in slightly different ways depending on the snap-option used. For each option they are explained in Sect. 12.4.1.

12.4 *Additional Pointers in SNAP*

A new set of pointers are introduced in order to address the additional pointers to the array geometry variables:

21 1	length of the array [only used with opt **o**]
21 2	bow of the parabola for the array [not with opt **x2** or **x3**]
21 3	rotation-1 (deg) [not with opt **x3**]
21 4	rotation-2 (deg) [not with opt **x3**]
21 5	rotation-3 (deg) [not with opt **x3**]

12.4.1 Examples of SNAP3D files

The examples here are in the snap3d directory in the examples directory. They are all for a horizontal array used in the SWellEx experiment.

Option o

This is the simplest option. Each receiver is uniformly distributed along the length of the array from "first receiver depth" (for $a = 0$) , i.e. for a equivalent straight array. The deflection of the array (x-coordinate) is then computed using a parabola along with the actual z-coordinates. For this option the "last receiver depth" in the receiver depth line are dummy.

For a simplified SWellEx environment the input file has the following format. The array is at 212.9 m depth and has 11 receivers uniformly spread over a length of 100 m with a bow of 15 m. The rotation angles are specified so that the array is horizontal. The SAGA option **C** defines that we are computing an ambiguity surface and are searching over the azimuth (rotation3) angle and the bow of the parabola.

```
TEST, opt o
C  c                ! options

 2000  64  10       ! niter, q, npop
 0.8 0.5 0.05  0    ! px,pu,pm
 o                  ! snap options
 13 500             ! freq  max_no_modes
49 64 79 94 112 130 148 166 201 235 283 338 388 100 200 388
 213.,0.,0.,0       ! water depth scatt(1),scatt(2),beta(0)
      0   1522
     10   1519
     20   1500
     30   1495
     61   1491
    213   1488
 30 1.76 0.2        ! sediment thickness, r1, beta(1)
 0   1572
 30  1593
 2.1, 0.06, 1880    ! bottom r2 beta(2),c2
 0,0                ! bottom shear beta(3) c2s
 60                 ! source depth
 212.9 212.9 11     ! rec depth: first, last, Nrec

100  15             ! array param: length[only with o], bow[not with x2,x3]
90 90 0             ! angles: rotation-1, rotation-2, rotation-3
 1                  ! number of ranges
 2140               ! receiver range

2                   ! nparm
```

```
 21 5 -50 100 150  ! azimuth [deg]
 21 2 -20  20  80  ! bow of parabola [m]
```

Option x

For this option the local z-coordinates of each receiver (for $a = 0$) are specified, i.e. for a equivalent straight array. The deflection of the array (x-coordinate) is then computed using a parabola along with the actual z-coordinates. The array is then rotated using the three Euler angles. For this option the array parameter [length] is dummy, as is the "last receiver depth" and in the receiver line. The x-coordinate in the input file is not used either.

The receiver block of the input file from *option o* is now modified so that 11 receivers are spread unevenly over a 74 m array. It has the following format for this option:

```
 212.9 212.9 11  0 ! rec depth

   0  0  ! z and x, the first element must be at (0,0)
   3  0
   7  0
  11  0
  15  0
  26  0
  32  0
  46  0
  54  0
  64  0
  74  0
100  15       ! array param: length[only with o], bow [not with x2, x3]
90 90 0             ! angles: rotation 1, rotation 2, rotation 3
```

Option x2

For this option the local x and z coordinates of each receiver are specified. The array is then rotated using the three Euler angles. For this option the array parameters [length and bow] are dummy, as is the "last receiver depth" in the receiver line.

The receiver block of the input file from *option o* is now modified so that 11 receivers are spread unevenly over a 74 m array and the x-coordinate is also specified. It has the following format for this option:

```
 212.9 212.9 11  0 ! rec depth

   0  0       ! z and x, the first element must be at (0,0)
   3  3
   7  4
  11  5
```

```
  15  5
  26  6
  32  6
  46  5
  54  3
  64  2
  74  0
100  15       ! array param: length[only with o], bow [not with x2, x3]
90 90 0             ! angles: rotation 1, rotation 2, rotation 3
```

Option x3

All the three coordinates of each element are directly specified (global z and local x and y). For this option all the array parameters [length, bow, angles] are dummy. The receiver depths can be uniformly spaced or unevenly spaced. Individual receiver depths are specified by specifying a $-$ Nrec in the receiver-depth-line.

The receiver block of the input file from *option o* is then modified so that first the depth of the array elements is specified from 10 to 170 m. The local x and y coordinates are then specified, here the array is at an angle of about 45^0 to the source-receiver line. It has the following format:

```
 212.9 212.9 -11  0 ! rec depth

10 20 40 60 90 100 110 130 140 150 170  ! the z depth
   0  0        ! x and y, the first element must be at (0,0)
   3  3
   7  8
  10  11
  15  16
  26  27
  32  33
  46  46
  54  53
  64  63
  74  73

100  15       ! array param: length[only with o], bow [not with x2, x3]
90 90 0             ! angles: rotation 1, rotation 2, rotation 3
```

References

[1] F.B. Jensen and M.C. Ferla, "SNAP: The SACLANTCEN normal-mode acoustic propagation model," SM-121, SACLANT Undersea Research Centre, La Spezia, Italy (1979).

[2] H. Schmidt, "SAFARI: Seismo-acoustic fast field algorithm for range independent environments. User's guide," SR-113, SACLANT Undersea Research Centre, La Spezia, Italy (1987).

[3] H. Schmidt, "OASES Version 1.6: Application and upgrade notes," Massachusetts Institute of Technology, Cambridge, MA (1993).

[4] D.D. Ellis, "A shallow water normal mode reverberation model," J. Acoust. Soc. Am. **97**, 2804–2814 (1995).

[5] F. Bini-Verona, P.L. Nielsen and F.B. Jensen, "PROSIM broadband normal-mode model: A user's guide," SM-358, SACLANT Undersea Research Centre, La Spezia, Italy, (1999).

[6] S.J. Levinson, E.K. Westwood, R.A. Koch, S.K. Mitchell and C.V. Sheppard, "An efficient and robust method for underwater acoustic normal mode computations," J. Acoust. Soc. Am. **97**, 1576–1585 (1995).

[7] E.K. Westwood, C.T. Tindle and N.R. Chapman, "A normal mode model for acousto-elastic ocean environments," J. Acoust. Soc. Am. **100**, 3631–3645 (1996).

[8] Dag Tollefsen, "User interface to ASSA and FGS with ORCA90" Norwegian Defence Research Establishment, FFI/NOTAT-2002/04824, Kjeller, Norge 2002

[9] E.K. Westwood, "ORCA Guide v3.0", ARL texas,2003

[10] E.K. Westwood, and P.J.Vidmar, "Eigenray finding and timeseries simulation in a layered-bottom ocean ," J. Acoust. Soc. Am. **81**, 912–924 (1987).

[11] E.K. Westwood, and C.T. Tindle, "Shallow water time series simulation using ray theory," J. Acoust. Soc. Am. **81**, 1752–1761 (1987).

[12] M.D. Collins "Users guide for RAM versions 1.0 and 1.0p", (ftp://ram.nrl.navy.mil/pub/RAM/), (2001)

[13] M.D. Collins, "A split-step Padé solution for parabolic equation method," J. Acoust. Soc. Am. **93**, 1726–1742 (1993).

[14] A.E. Barrios, "Parabolic equation modeling in horizontally inhomogeneous environments," IEEE Trans. on Ant. and Prop. **36**, 791–797 (1992).

[15] A.E. Barrios, "A terrain parabolic equation model for propagation in the troposphere," IEEE Trans. on Ant. and Prop. **42**, 90–98 (1994).

[16] P. Gerstoft, "Inversion of seismoacoustic data using genetic algorithms and *a posteriori* probability distributions," J. Acoust. Soc. Am. **95**, 770–782 (1994).

[17] Purnima Ratilal, P. Gerstoft J.T. Goh and Keng Pong Yeo, "Inversion of pressure data on a vertical array to determine the seafloor geoacoustic properties," J. of Computational Acoustics (June 1998).

[18] P. Gerstoft, "Inversion of acoustic data using a combination of genetic algorithms and the Gauss–Newton approach," J. Acoust. Soc. Am. **97**, 2181–2190 (1995).

[19] M.D. Collins, W.A. Kuperman and H. Schmidt, "Nonlinear inversion for ocean-bottom properties," J. Acoust. Soc. Am. **92**, 2770–2783 (1992).

[20] L. Ingber, "Very fast simulated reannealing," Mathematical Computer Modeling **12**, 967–973 (1989).

[21] L. Ingber, "Simulated annealing: Practice versus theory," Mathematical Computer Modeling **18**, 29–57 (1993).

[22] A. Caiti, T. Akal and R.D. Stoll, "Estimation of shear wave velocity in shallow marine sediments," IEEE J. Oceanic Eng. **19**, 58–72 (1994).

[23] P. Gerstoft and A. Caiti, "Acoustic estimation of bottom parameters: error bounds by local and global methods," in *Second European Conference on Underwater Acoustics*, edited by L. Bjørnø (European Commission, Luxembourg, 1994), pp. 887–892.

[24] F.B. Jensen, W.A. Kuperman, M.B. Porter and H. Schmidt, *Computational Ocean Acoustics* (American Institute of Physics, New York, 1994).

[25] A. Tolstoy, *Matched Field Processing for Underwater Acoustics* (World Scientific, Singapore, 1993).

[26] S.M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall, Englewood Cliffs, NJ, 1993).

[27] Y. Bard, *Nonlinear Parameter Estimation* (Academic Press, San Diego, CA, 1974).

[28] W. Menke, *Geophysical Data Analysis: Discrete Inverse Theory* (Academic Press, San Diego, CA, 1989).

[29] R.A. Parker, *Geophysical Inverse Theory* (Princeton University Press, Princeton, NJ, 1994).

[30] A. Tarantola, *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation* (Elsevier, Amsterdam, 1987).

[31] M.K. Sen and P.L. Stoffa, *Global Optimization in Geophysical Inversion* (Elsevier, Amsterdam, 1995).

[32] M.D. Collins and W.A. Kuperman, "Focalization: Environmental focusing and source localization," J. Acoust. Soc. Am. **90**, 1910–1922 (1991).

[33] C.E. Lindsay and N.R. Chapman, "Matched field inversion for geophysical parameters using adaptive simulated annealing," IEEE J. Oceanic Eng. **18**, 224–231 (1993).

[34] S.E. Dosso, M.L. Yeremy, J.M. Ozard and N.R. Chapman, "Estimation of ocean bottom properties by matched-field inversion of acoustic field data," IEEE J. Oceanic Eng. **18**, 232–239 (1993).

[35] Peter Gerstoft and Zoi-Heleni Michalopoulou "Global Optimization in Matched-Field Inversion" *Fourth European conference on underwater acoustics*, pp 27–32, Rome 1998.

[36] J.A. Scales, M.L. Smith and T.L. Fisher, "Global optimization methods for highly nonlinear inverse problems," J. Comp. Phys. **103**, 258–268 (1992).

[37] M. Sambrigde and G. Drijkoningen, "Genetic algorithms in seismic waveform inversion," Geophys. J. Int. **109**, 323–343 (1992).

[38] P.L. Stoffa and M.K. Sen, "Multiparameter optimization using genetic algorithms: Inversion of plane wave seismograms," Geophysics **56**, 1794–1810 (1991).

[39] M.K. Sen and P.L. Stoffa, "Rapid sampling of model space using genetic algorithms: examples from seismic waveform inversion," Geophys. J. Int. **108**, 281–292 (1992).

[40] S.M. Jesus and A. Caiti, "Estimating geoacoustic bottom properties from towed array data," J. Comp. Acoust. **4**, 273–290 (1996).

[41] Z.-H. Michalopoulou, H. Martynov and M.B. Porter, "Simulated annealing and genetic algorithms for broadband source localization," in *Third European Conference on Underwater Acoustics*, edited by J.S. Papadakis (Crete University Press, Crete, 1996), pp. 409–414.

[42] O. Lotsberg and S.M. Jesus, "Matched field inversion of geoacoustic properties from towed array data in shallow water," in *Third European Conference on Underwater Acoustics*, edited by J.S. Papadakis (Crete University Press, Crete, 1996), pp. 601–606.

[43] M.J. Rendas and G. Bienvenu, "Tuning genetic algorithms for underwater acoustics using *a priori* statistical information," Proc. IEEE ICASSP-97, Munich (1997).

[44] J. Heitkötter, D. Beasley, "The hitch hikers guide to evolutionary computing" (FAQ for comp.ai.genetic) (2000)

[45] P. Gerstoft and C.F. Mecklenbräuker, "Ocean acoustic inversion with estimation of *a posteriori* probability distributions," J. Acoust. Soc. Am. **104** 808–819 (1998).

[46] M.B. Porter and A. Tolstoy, "The matched field processing benchmark problems," J. Comp. Acoust. **2**, 161–185 (1994).

[47] D.F. Gingras and P. Gerstoft, "Inversion for geometric and geoacoustic parameters in shallow water: Experimental results," J. Acoust. Soc. Am. **97**, 3589–3598 (1995).

[48] D.F. Gingras, P. Gerstoft and N.L. Gerr, "Electromagnetic matched field processing: Basic concepts and tropospheric simulations," to appear IEEE Trans. on Ant. and Prop. (1997).

[49] H.G. Schneider, "Acoustic models at SACLANTCEN," SM-285, SACLANT Undersea Research Centre, La Spezia, Italy (1995).

[50] A.B. Baggeroer, W.A. Kuperman, and H. Schmidt, "Matched field processing: Source localization in correlated noise as an optimum parameter estimation problem," J. Acoust. Soc. Am. **83**, 571-587 (1988).

[51] B. Ottersten, M. Viberg, P. Stoica and A. Nehorai, "Exact and Large sample maximum likelihood estimation and detection in Array processing," in *Radar array processing* edited by Haykin, Litva and Shepherd (Spinger Verlag, Berlin, 1993).

[52] P. Gerstoft, "Global inversion by genetic algorithms for both source position and environmental parameters," J. Comp. Acoust. **2**, 251–266 (1994).

[53] C.H. Harrison and J.A. Harrison, "A simple relationship between frequency and range averages for broadband sonar," J. Acoust. Soc. Am. **97**, 1314–1317 (1995).

[54] M. Lambert, "Inversion of seismo-acoustic real data using genetic algorithms," SM-276, SACLANT Undersea Research Centre, La Spezia, Italy (1994).

[55] P. Gerstoft and J.T. Goh, "Performance evaluation of horizontal and vertical vector sensor arrays in shallow water environments," Proceedings 16th International Conference on Acoustics, p 1643–1644 (1998).

[56] J.-P. Hermand and P. Gerstoft, "Inversion of broadband multitone acoustic data from the Yellow Shark summer experiments," IEEE J. Oceanic Eng. **21**, 324–346 (1996).

[57] D.J. Thompson, "Jacknifing using multiple-window spectra," Proc. IEEE ICASSP-94, Adelaide (1994).

[58] C.F. Mecklenbräuker, D. Maiwald and J.F. Böhme, "F-test in matched field processing: identifying multimode propagation", Proc. IEEE ICASSP-95, Detroit (1995).

[59] P. Gerstoft and D.F. Gingras, "Parameter estimation using multi-frequency range-dependent acoustic data in shallow water," J. Acoust. Soc. Am. **99**, 2839–2850 (1996).

[60] G. Haralabus and P. Gerstoft, "Stability of parameter estimates using multi-frequency inversion techniques," SR–253, SACLANT Undersea Research Centre, La Spezia, Italy (1996).

[61] G. Haralabus and P. Gerstoft, "Variability of multi-frequency parameter estimates in shallow water," in *Third European Conference on Underwater Acoustics*, edited by J.S. Papadakis (Crete University Press, Crete, 1996), pp. 355–360.

[62] N.R. Chapman and A. Tolstoy, *Geoacoustic Benchmark Inversion Workshop*, J. of Computational Acoustics (Oct 1998)

[63] M. Siderius, P. Gerstoft and P.L. Nielsen, "Broadband acoustic inversion using a sparse array," J. of Computational Acoustics (Oct 1998)

[64] D.D. Ellis and P. Gerstoft, "Using inversion techniques to extract bottom scattering strengths and sound speed from shallow water reverberation data," in *Third European Conference on Underwater Acoustics*, edited by J.S. Papadakis (Crete University Press, Crete, 1996), pp. 887–892.

[65] H.V. Hitney, "Refractive effects from VHF to EHF: Part A, Propagation mechanisms," AGARD Lecture Series, LS–196 (1994).

[66] J.B. Marion and S.T. Thornton, "Classical dynamics of particles and systems," (Harcourt Brace Jovanovich, San Diego, 1988)

[67] A. J. W. Duijndam, "Bayesian estimation in seismic inversion. Part I: Principles," Geophysical Prospecting **36**, 878–898 (1988).

[68] J. Lefebvre, H. Roussel, E. Walter, D. Lecointe and W. Tabbara, "Prediction from wrong models: The Kriging approach," IEEE Ant. Prop. Magazine, **38**(4), 35–45 (1996).

[69] D. Maiwald and J.F. Böhme, "Multiple testing for seismic data using bootstrap," Proc. IEEE ICASSP-94, **6**, Adelaide (1994), pp. 89–92.

[70] M.L. Hinich, "Maximum likelihood estimation of a radiating source in a waveguide", J. Acoust. Soc. Am. **66**, 480–483 (1977).

[71] E.C. Shang, "Source depth estimation in wave guides," J. Acoust. Soc. Am. **86**, 1960–1964 (1985).

[72] S. Rajan, "Waveform inversion of geoacoustic parameters in the ocean bottom," J. Acoust. Soc. Am. **95**, 3228–3241 (1992).

[73] P. Gerstoft, D.F. Gingras, L.T. Rogers and W.S. Hodgkiss, "Estimation or radio refractivity structure using matched field array processing," IEEE antenna and Propagation, **48** 345–356 (2000).

[74] Peter Gerstoft, William S. Hodgkiss, W.A. Kuperman and Heechun Song "Phenomenological and global optimization inversion," IEEE oceanic Eng. Special issue geoacoustic inversion, to appear April 2003

[75] C. Park, W. Seong, P.Gerstoft, M.Siderius "Timedomain geoacoustic inversion of high frequency chirps using a towed system" IEEE oceanic Eng. Special issue geoacoustic inversion. To appeAr April 2003.

[76] Edwin L. Hamilton, "Geoacoustic modeling of the sea floor," J. Acoust. Soc. Am. **68**, Issue 5, pp. 1313-1340 (1980)

[77] Richard T. Bachman, "Estimating velocity ratio in marine sediment," J. Acoust. Soc. Am. **86**, Issue 5, 2029–2032 (1989)

Annex A

Likelihood functions for a vertical array

Depending on the model used for the error or noise distribution of the data, a specific likelihood function must be used. Thus prior knowledge about the error function is required. Usually this is not directly available and some simple and reasonable models must be used. Two special cases are considered for multi-frequency vertical array data. The noise distribution on each phone is assumed complex Gaussian. First, in Sect. A.1 it is assumed that the noise is independent on each phone and, second, in Sect. A.2 it is assumed independent for each *significant* mode.

Often, a distinction is made between errors due to noise in the data and errors due to an incomplete forward model, because neither the theory nor the environmental model is adequate. If both error types belong to the same distribution, there is no reason to consider them separately [67]. Here only one error term is considered.

Recently, there has been progress in describing both errors using Kriging [68], in which errors due to noise and incomplete forward modeling are considered separately. Both noise and modeling errors are assumed zero-mean Gaussian and independent. The modeling errors are assumed to possess a given correlation structure depending on the "distance" between two environmental models. This correlation structure is chosen empirically. Clearly, the same value of the model parameters should correspond to the same value of the modeling error.

A.1 Multi-frequency matched field processing

The relation between the observed complex-valued data vector $\mathbf{q}(\omega)$ on an N-element hydrophone antenna and the predicted data $\mathbf{p}(\mathbf{m}, \omega)$ at a angular frequency ω is described by the model

$$\mathbf{q}(\omega) = \mathbf{p}(\mathbf{m}, \omega) + \mathbf{e}(\omega) \ , \tag{43}$$

where $\mathbf{e}(\omega)$ is the error term. The predicted data is given by $\mathbf{p}(\omega) = \mathbf{w}(\mathbf{m}, \omega)S(\omega)$, where the complex deterministic source term $S(\omega)$ is unknown. The transfer function $\mathbf{w}(\mathbf{m}, \omega)$ is obtained using an acoustic propagation model and an environmental model $\mathbf{m}$.

The errors are assumed to be additive; they stem from many sources: errors in describing the environment, errors in the forward model, instrument and measurements errors, and noise in the data. For the predicted acoustic field "reasonably close" to the true field, this error term is assumed complex Gaussian distributed, stationary with zero mean and diagonal covariance matrix $\nu(\omega)\mathbf{I}$, where the error power spectrum ν is unknown. Or written compactly $\mathbf{e}(\omega) \sim \mathrm{CN}(0, \nu(\omega)\mathbf{I})$, where CN symbolizes the Complex Gaussian distribution. Thus, the data $\mathbf{q}(\omega)$ on the receiving array are also complex Gaussian distributed, $\mathbf{q}(\omega) \sim \mathrm{CN}\left(\mathbf{p}(\omega, \mathbf{m}), \nu(\omega)\mathbf{I}\right)$. For the derivation of a maximum likelihood estimate, it is further assumed that the data for each frequency bin are uncorrelated across frequency and time snapshot and that for each time snapshot the source term $S(\omega)$ can vary whereas the error power $\nu(\omega)$ is constant.

In the following, we will often abbreviate $\mathbf{q}_l = \mathbf{q}(\omega_l)$, where $\{\omega_l | l = 1, \dots, L\}$ is a suitable set of frequency bins. We have $\mathsf{E}\mathbf{q}_l\mathbf{q}_l^\dagger = \mathbf{R}_l = \mathbf{p}_l\mathbf{p}_l^\dagger + \nu_l\mathbf{I}$. Under these assumptions the likelihood function evaluates to

$$\mathcal{L} \propto \prod_{l=1}^{L} \frac{1}{\nu_l^N} \exp\left(-\frac{\phi_l}{\nu_l}\right) , \tag{44}$$

where[2]

$$\phi_l = \mathrm{tr}\hat{\mathbf{R}}_l - \frac{\mathbf{w}_l^\dagger \hat{\mathbf{R}}_l \mathbf{w}_l}{\mathbf{w}_l^\dagger \mathbf{w}_l} . \tag{45}$$

Optimization for ν_l yields the closed form ML solution

$$\hat{\nu}_l = \frac{1}{N}\phi_l . \tag{46}$$

The $N \times N$ Hermitian matrix $\hat{\mathbf{R}}_l$ denotes the estimated cross-spectral density matrix of the observed data "in phone-space", see Sect. A.3. With these definitions, the ML objective function can be written as

$$\phi = \prod_{l=1}^{L} \phi_l = \prod_{l=1}^{L} \left(\mathrm{tr}\hat{\mathbf{R}}_l - \frac{\mathbf{w}_l^\dagger \hat{\mathbf{R}}_l \mathbf{w}_l}{\mathbf{w}_l^\dagger \mathbf{w}_l}\right) . \tag{47}$$

Using a global optimization procedure, the minimum $\hat{\phi}^{\mathrm{ML}}$ for the ML solution $\hat{\mathbf{m}}^{\mathrm{ML}}$ is estimated. The estimate, Eq. (46), is biased. The bias stems from the degrees of freedom in the estimated parameters: source signal S and nonlinear parameters $\mathbf{m}$, see [69]. For simplicity this bias is neglected here. As the noise power spectral density now is estimated, Eq. (46), the likelihood function is given by

$$\begin{aligned} \mathcal{L}(\mathbf{m}) &= p(\mathbf{m}|\mathbf{q}) \\ &\propto \prod_{l=1}^{L} (\hat{\nu}_l{}^{\mathrm{ML}})^{-N} \exp\left(-\frac{\phi_l(\mathbf{m})}{\hat{\nu}_l^{\mathrm{ML}}}\right) \\ &\propto \prod_{l=1}^{L} \exp\left(-N\frac{\phi_l(\mathbf{m}) - \hat{\phi}_l^{\mathrm{ML}}}{\hat{\phi}_l^{\mathrm{ML}}}\right) . \end{aligned} \tag{48}$$

[2] † refers to the Hermitian transpose

The problem, as addressed above, is then to integrate this multi-dimensional probability distribution. Often this integral can be evaluated with sufficient accuracy using the information from the global search. In some cases it might be necessary to increase the sampling of the model space in order to obtain convergence.

According to Eq. (48), the likelihood function has a stronger maximum when more hydrophones are used. When inverting observed data, there is a limit to how much useful information can be obtained by adding additional hydrophones, as they then become strongly correlated. At high SNR it is expected that the main error contribution is due to inadequate forward modeling. Further, the number of uncorrelated hydrophones is approximately the same as the number of propagating modes, because this limits the degrees of freedom in the random part of the acoustic wave field. The number of uncorrelated hydrophones is estimated as the rank of the covariance matrix.

A.2 Multi-frequency matched mode processing

Normal modes provide a complete description of the field at long ranges, and thus one can equivalently process the data in phone-space or in mode-space. The matched mode approach is described by Tolstoy [25], Hinich [70], and Shang [71]. Modal processing is discussed here as an alternative noise estimate when there are more hydrophones than propagating modes.

We assume that the observed field of N sensors can be approximately expressed via a set of J *significant* modes, expressed in a $N \times J$ matrix $\mathbf{V}(\omega) = (\mathbf{v}_1, \ldots, \mathbf{v}_J)$, and the corresponding complex valued modal amplitudes[3] $\breve{\mathbf{q}}(\omega) = (\breve{q}_1, \ldots, \breve{q}_J)'$ are:

$$\mathbf{q}(\omega) \approx \sum_{j=1}^{J} \mathbf{v}_j \breve{q}_j = \mathbf{V}(\omega)\breve{\mathbf{q}}(\omega) \ . \tag{49}$$

It is assumed that we have more hydrophones than modes. We can invert this relationship in a least-squares sense and estimate the vector of modal amplitudes $\breve{\mathbf{q}}_l = \breve{\mathbf{q}}(\omega_l)$ in mode-space from the observation $\mathbf{q}_l = \mathbf{q}(\omega_l)$ in phone-space,

$$\breve{\mathbf{q}}_l = (\mathbf{V}_l^\dagger \mathbf{V}_l)^{-1} \mathbf{V}_l^\dagger \mathbf{q}_l \ . \tag{50}$$

Note that the modes and modal amplitudes depend on the environment. When optimizing the environment $\mathbf{m}$, the modal amplitudes will change with the environment.

We assume a simple relationship between the observed modal amplitudes $\breve{\mathbf{q}}_l$ and synthetic generated modal amplitudes,

$$\breve{\mathbf{q}}_l = \breve{\mathbf{p}}_l(\mathbf{m}) + \breve{\mathbf{e}}_l(\mathbf{m}) \ , \tag{51}$$

[3] The symbol ˘ refers to the mode-space.

where $\breve{\mathbf{p}}_l(\mathbf{m}) = S_l \breve{\mathbf{w}}_l(\mathbf{m})$ is the complex-valued modal amplitudes of the synthetic data and $\breve{\mathbf{e}}_l$ represents the error term for each mode. It is assumed that the noise covariance matrix is diagonal for the J significant modes and the noise power $\breve{\nu}$ is identical for all J modes.

Using a similar approach to that in Sect. A.1, the objective function is

$$\breve{\phi}_l = \mathrm{tr}\hat{\breve{\mathbf{R}}}_l - \frac{\breve{\mathbf{w}}_l^\dagger \hat{\breve{\mathbf{R}}}_l \breve{\mathbf{w}}_l}{\breve{\mathbf{w}}_l^\dagger \breve{\mathbf{w}}_l} \ , \tag{52}$$

where $\hat{\breve{\mathbf{R}}}_l$ is the estimated covariance matrix between the modes. But using the expression for the modes, Eq. (49), the objective function in mode-space, Eq. (52), is seen to be equivalent to the objective function in phone-space, i.e. Eq.(45) expressed in phone-space,

$$\breve{\phi}_l \approx \phi_l \ . \tag{53}$$

The noise estimate is obtained using the approximation in Eq. (49),

$$\hat{\breve{\nu}}_l = \frac{1}{J}\hat{\breve{\phi}}_l^{\mathrm{ML}} \approx \frac{1}{J}\hat{\phi}_l^{\mathrm{ML}} \ . \tag{54}$$

The likelihood function becomes

$$\begin{aligned}\mathcal{L}(\mathbf{m}) = p(\mathbf{m}|\mathbf{q}) \quad &\propto \quad \prod_{l=1}^{L} (\hat{\breve{\nu}}_l^{\mathrm{ML}})^{-J} \exp\left(-\frac{\breve{\phi}_l(\mathbf{m})}{\hat{\breve{\nu}}_l^{\mathrm{ML}}}\right) \\ &\propto \quad \prod_{l=1}^{L} \exp\left(-J\frac{\phi_l(\mathbf{m}) - \hat{\phi}_l^{\mathrm{ML}}}{\hat{\phi}_l^{\mathrm{ML}}}\right) \ .\end{aligned} \tag{55}$$

The advantage of this formulation is that it does not depend directly on the number of hydrophones, but only on the number of propagating modes. For many hydrophones ($N \gg J$) this likelihood function seems more realistic. The precise value of J is not yet clear. For simplicity, J is assumed to be independent of frequency. Only the objective function is affected by the choice of J. All the propagating modes are incorporated in the forward model.

A.3 Estimation of the covariance matrix

In order to estimate the covariance matrix $\mathbf{R}_l$, the received time signal is divided into K time frames. Each frame was short-time Fourier transformed using the multiple-windows technique described in Refs. [57, 58],

$$\mathbf{q}_{k,p}(\omega) = \sum_{t=0}^{T-1} \nu_t^p \mathbf{q}(t+kT)\mathrm{e}^{-j\omega t}, \qquad \text{for} \quad \begin{cases} k = 0, \ldots, K-1 \\ p = 0, \ldots, P-1 \end{cases} \tag{56}$$

where ν^p is a special set of P orthonormal data tapers [57, 58]. The correlation matrix $\mathbf{R}$ was estimated at each selected frequency ω_l as the ensemble average, i.e.

$$\hat{\mathbf{R}}(\omega_l) = \frac{1}{KP} \sum_{k=0}^{K-1} \sum_{p=0}^{P-1} \mathbf{q}_{k,p}(\omega_l) \mathbf{q}_{k,p}^{\dagger}(\omega_l). \tag{57}$$

In order to obtain a good estimation of the noise, it is required that $KP \gg N$, where N is the number of hydrophones. In order to "just" estimate the signal and the unknown parameters $\mathbf{m}$, the number of averages, KP, can be much smaller for a received signal with sufficient SNR.

Annex B
Regularization

Regularization was originally associated with the Thikhonov technique for filtering out the high frequencies of a solution. Today it is used in a wider sense; here we will understand it as an approach that restricts parts of the inverse solution.

It is well-known that the numerical solutions to inverse problems are often ill-conditioned. One remedy to ill-conditioning is regularization. The idea of regularization is to introduce *a priori* knowledge on the physical solution. This can be done in several ways:

1) Reparameterization – shape functions, see Sect. 9.
2) Thikhonov regularization. See Sect. B.1 below.
3) Weighting the likelihood function with the *a priori* distribution. See Sect. B.2.

B.1 Thikhonov regularization

Given the *a priori* estimate $\mathbf{m}_0$ of the solution it seems natural to minimize the difference between the solution $\mathbf{m}$ and the estimate $\mathbf{m}_0$

$$\phi_{\text{reg}} = \|L(\mathbf{m}_0 - \mathbf{m})\| \; . \tag{58}$$

Normally, L would be the identity matrix or a discrete approximation to a derivative operator. Due to different physical dimensions of the parameters, an *ad hoc* regularization could be

$$\phi_{\text{reg}} = 1/N_{\text{parm}} \sum \frac{|m_i - m_i^{apri}|^\gamma}{|m_i^{\max} - m_i^{\min}|^\gamma} \; , \tag{59}$$

where m_i^{apri} is the *a priori* expected value, $m_i^{\max} - m_i^{\min}$ is the search interval, and the exponent γ controls the shape of the penalization. We use $\gamma = 1$.

The objective function combining the *a priori* knowledge and the data is then

$$\phi = \phi_B + \lambda\phi_{\text{reg}}, \tag{60}$$

where λ is a Lagrange multiplier. We use $\lambda = 1$ and thus place equal emphasis on the data and the *a priori* information.

For option $\mathbf{L}$ we have implemented Eq. (60) with $\lambda = 1$ and $\gamma = 1$.

B.2 Including the *a priori* probability distribution

Usually, when solving inverse problems, the question is "What is the environmental model for this given data set?" This is normally an ill-posed question. A better question is "What can be inferred from the data about the environmental model given some environmental information?" Thus some *a priori* information should be included in the inverse problem. *A priori* information is always used in global inversion schemes. The model structure is selected based on *a priori* knowledge and uniform *a priori* distributions are used between the minimum and maximum bounds for the parameters.

One possibility is to include the *a priori* model in the objective function, see e.g. [30, 72]. This has the disadvantage that the distribution must be known explicitly. For Gaussian *a priori* distribution the objective function consists of two terms, one measuring the match between observed and synthetic data and the second penalizing the deviation from the *a priori* model. This approach is used in linearized inversions in order to regularize the solution.

Here a simple approach is used: the obtained likelihood function is multiplied with the *a priori* distribution, according to Eq. (5). This distribution can be arbitrary, for example a smoothed distribution obtained from inversion of other data. A simple triangular distribution is often used:

$$\rho(m^i) \propto \begin{cases} (m^i - m_u^i)/(m_m^i - m_u^i) & \text{for } m_m^i < m^i < m_u^i \\ (m^i - m_l^i)/(m_m^i - m_l^i) & \text{for } m_l^i < m^i < m_m^i \\ 0 & \text{otherwise} \end{cases} \tag{61}$$

where $m_l^i < m_u^i < m_m^i$ are the abscissa of lower bound, maximum, and upper bound of the *a priori* distribution, respectively.

A priori information is also used in the parameterization of the forward model. The choices made when doing this have a significant influence on the inverse solution, probably more than including *a priori* information for each parameter. In discretizing the environment, the physics should be carefully considered and described efficiently. Shape functions [18] is a useful method to obtain an efficient description which provide a mapping between the environmental model and the numerical forward model. This could, for example, be used to limit the search to only positive gradients in the sediment, or to obtain a more efficient description of the environment.

It is assumed that there is vanishing correlation between the *a priori* distributions of the individual parameters. Thus, $\rho(\mathbf{m}) = \rho_1(m^1)\rho_2(m^2)\cdots$. Otherwise the distributions become too complicated. If parameters are correlated, the search can be limited by using a correlated *a priori* distribution. However, this seems too complicated and shape functions are used instead to map a correlated model vector to another representation with a lower degree of correlation.

Annex C
Updating SAGA

It is expected that the user may wish to update the SAGA program to invert other types of data. This may require inclusion of other forward modeling codes into the SAGA package, or a change in the objective function. This appendix provides guidance on the implementation of some of these changes. The author is also willing to help in this process.

C.1 Porting a forward model into SAGA

The following steps are necessary for porting a forward code (here called fm) to SAGA:

1. In order to do inversion with a new program it should be reliable. The dimensions of the matrices should be automatically checked. It should not crash. It should have an execution time of around one second.

2. Change the forward code to a subroutine. It should have the following structure, using the same subroutine names:

```
program sagaporting
call input
call forwardinit
do i=1,10
     call forw2
enddo
end
```

3. All the input should be read in one subroutine. The variables should be transferred to a computational subroutine via common blocks that are located in one include file. (For SAGA it is useful if this routine only contains a few variables, so that the declared variables in `fm` does not interfere with SAGA variables).

4. The computational subroutine should not have any internal read/write statements. Informational write statements are useful, but these should be lumped together and surrounded by an `if (flagpu .gt. number) then` statement [`flagpu` is an integer variable in a common block that is updated at each call].

5. It should be possible to call the computational subroutine several times. Some initialization of the forward program might only be necessary once, provided the variables are stored in common blocks. These initializations should be moved to the start of the subroutine `forwardinit`. An entry call `forw2` can then be inserted below these lines for subsequent calls.

6. The output of the program should be stored to a variable `resp`. It has the following structure:

```
do i_freq=1,M_freq
    do i_depth=1,M_depth
        i_pointer=(i_depth-1+ (i_freq-1)*M_depth)*M_range
        do i_range=1,M_range
            resp(i_range+i_pointer)=press(i_range,i_depth,i_freq)
        enddo
    enddo
enddo
```

7. Check that `resp` contains precisely the same results when the forward model is called twice.

Only when the stand-alone version is working well is it time to interface it with the SAGA modules. It is much easier to work with the forward model alone, so it is important that all of the above works well before interfacing with SAGA.

The interface between SAGA and the forward model is done through two subroutines, `fm_init` that contains the `input` subroutine and the `forwardinit` subroutine, and `fm_inter` that transfer the SAGA-pointers to physical variables in the forward model.

1. Make a list of the variables that could be optimized and assign a number to each variable. During the optimization the variables are addressed through a pointer called `par2phy`. If the variable is a vector then the `index` is addressed by the one pointer `par2lay`. If the variable is a matrix it is indexed by two pointers, `par2lay` and `par3`, for the first and second index, respectively.

2. Copy one of the *inter.f files from one of the other forward models in /src to /src/fm_inter.f and modify this file according to the list of variables.

3. Now the forward model is ready to be ported into SAGA: copy the program in a directory below the saga/src directory, i.e. /fm.

4. Copy one of the *init.f files from one of the other forward models in /src to /src/fm_init.f and modify this file. The input subroutine should be called `input`. The main forward subroutine is called `forwardinit`. The entry statement is called `forw2`. Carefully change the copied file into the one used for the present forward model.

5. Update the `makefile` and compile.

C.2 Introducing a new option

It is quite easy to introduce a new option into SAGA. First find the subroutine `gaoptions` in file `gasub.f`. Find an unused letter and a corresponding value of the pointer `iopt`. Most new options will probably only require modification in a particular subroutine. All computations related to the objective function are done in subroutine `cost.f`.