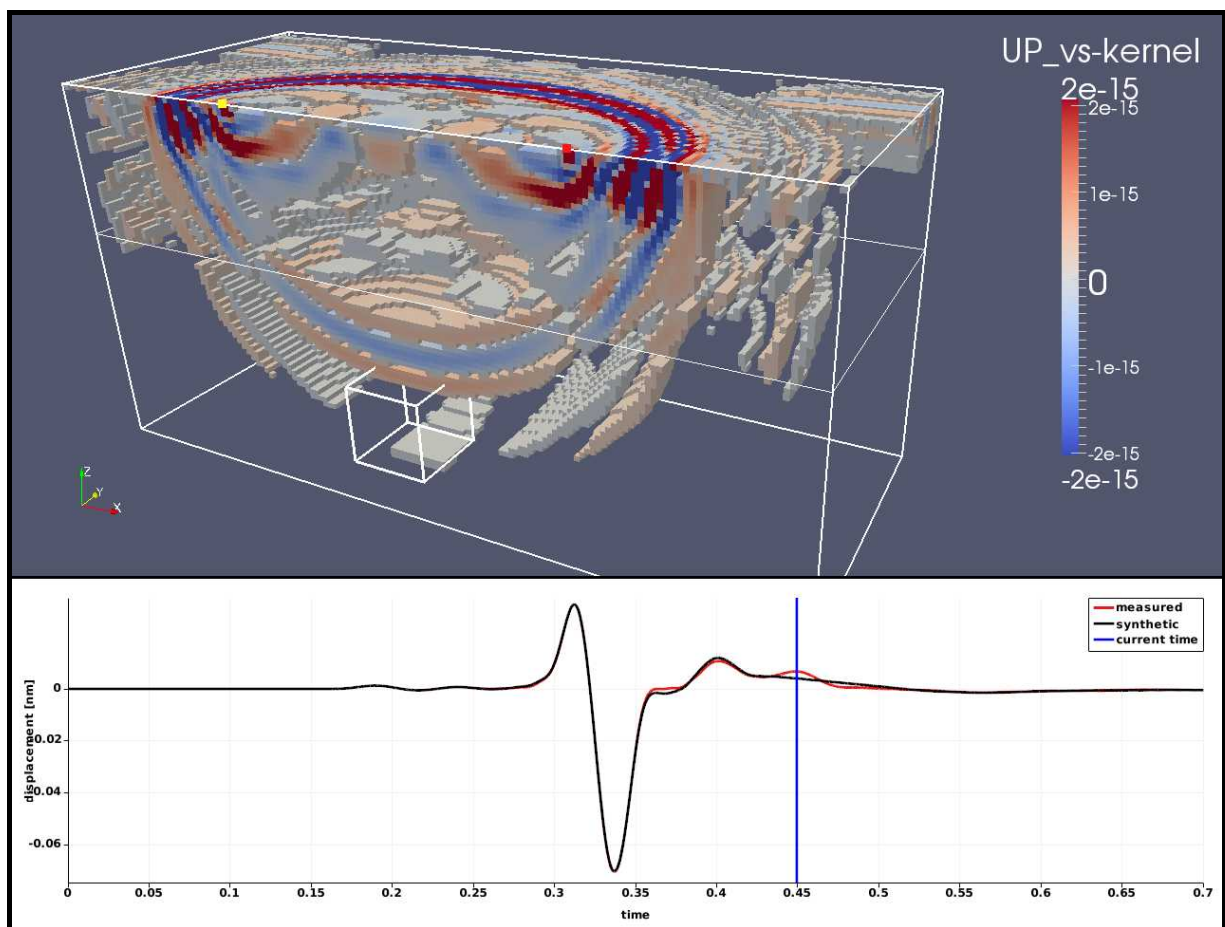


ASKI

Analysis of
Sensitivity and
Kernel
Inversion

User Manual
ASKI – version 1.0

Dec 2015
Florian Schumacher
Ruhr-Universität Bochum, Germany



Copyright ©2015 Florian Schumacher. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you use `ASKI` for your own research, please cite our paper [SFL 8]:

F. Schumacher, W. Friederich and S. Lamara,

”A flexible, extendable, modular and computationally efficient approach to scattering-integral-based seismic full waveform inversion”,

Geophysical Journal International,

Accepted 2015 November 17. Received 2015 November 17; in original form 2015 April 8

<http://dx.doi.org/10.1093/gji/ggv505>

This documentation was written in the hope that it will be useful to the user, but it *cannot be assured* that it is accurate in every respect or complete in any sense. In fact, at some places *this manual is work in progress*.

lease do not hesitate to report any inconsistencies via <http://www.rub.de/aski> or to improve this documentation by incorporating your experiences with `ASKI` and your personal experience of getting used to it (plus, let us know about it! Thanks). When you have developed new `ASKI` components or have modified existing ones, please extend / modify this document accordingly.

Furthermore, I am aware of the poor \LaTeX coding of this document. There is a lot of potential to improve the document style, hence the readability of the manual as a whole, as well as the coding style of the particular `.tex` files. *Please do not hesitate to improve!*

The \LaTeX source files and all related components of this document are available via

<http://www.rub.de/aski>

Florian Schumacher, Dec 2015

How to use this manual

Only chapter Guide (page 11) is intended to be read through which, for this reason, is held as compact as possible. This chapter may itself be regarded as “the manual”, with the appending chapters only containing more specific detail on processes or objects which chapter Guide refers to. After all, the very modular character of `ASKI` requires documentation which itself is modular.

In other words: just start reading the respective section of chapter Guide, which you are interested in and whenever you feel the need for more detail follow the respective references. This way, we try to focus the user on necessary information and successfully guide through the lot of details contained in this document.

When you conduct a specific `ASKI` operation for the first time, we recommend you to first fully read through the respective guiding list and the referred basic steps before you start running any programs. This way you will get an impression of the requirements for your operation.

All chapters appending chapter Guide are not intended to be read through section by section but may well serve the user as a reference.

What is ASKI ?

ASKI is a modularized software package written by the main authors Florian Schumacher and Wolfgang Friederich (Ruhr-University Bochum, Germany). It offers sensitivity and regularization analysis tools for seismic datasets as well as a scattering-integral-type full waveform inversion (FWI) concept based on waveform sensitivity kernels (i.e. Fréchet derivatives of seismic waveform data functionals) derived from Born scattering theory, having Gauss-Newton convergence properties. This inversion concept is presented in our paper [SFL 8] and in F. Schumachers doctoral dissertation [FS06].

ASKI does not implement an intrinsic code for simulation of seismic wave propagation in order to solve the forward problem, but instead provides generalized interfaces to different external wave propagation codes. At the moment, the 1D semi-analytical code GEMINI [FD95] and the 3D spectral element code SPECFEM3D [TKL08] are supported in *both*, Cartesian and spherical framework. Additionally ASKI supports the 3D nodal discontinuous Galerkin code NEXD (Lasse Lambrecht, Ruhr-University Bochum, Germany) and we will implement support for the Finite Difference code SOFI (KIT Karlsruhe, Germany) in the near future. Please do not hesitate to add your own forward code, we are happy to support you during that process.

The resulting very modular concept of ASKI allows us to keep the spatial description of the inverted models completely independent of the spatial description of the model used for solving the forward problem. This way, the overall iverse problem can be approached and regularized more physically compared with using the forward grid also for inversion. The normally very fine spatial discretization of the model domain is employed by the forward solver. This discretization is usually weigh too fine to be used directly for inversion since usually seismic data can only resolve much coarser structures and the resolution also varies throughout the model domain. The separation concept of ASKI *naturally* allows to choose the spatial model resolution heterogeneously and *anew* in each iteration of FWI.

Instead of using time-dependent values of ground motion (i.e. samples of a time-series of seismic data), ASKI uses frequency-dependent complex values of ground motion, recorded at a certain receivers excited by a certain seismic sources. This, mainly, has reasons of computational feasibility. For time-domain forward methods, we implement an on-the-fly Fourier Transform in order to provide the required spectral waveforms.

Using sensitivity kernels K , data residuals δd_i are connected to model updates δm by an integral relation $\delta d_i = \int_{\text{earth}} \delta \vec{m} \cdot \vec{K}_i$. In order to build a linear system, the model update $\delta \vec{m}$ is assumed to be constant throughout small scattering volumes Ω_j , where $\Omega = \dot{\bigcup}_j \Omega_j$. These volumes constitute the cells of the volumetric inversion grid and the sensitivity matrix contains entries of preintegrated kernels $\int_{\Omega_j} K_i$.

The sensitivity kernels K are computed from forward wavefields and strains produced on a set of points in space, which is dependent on the particular forward method. This set of points is referred to as *wavefield points*. The wavefields are written to file, by the respective forward method,

which may require large discspace. Providing methods for constructing quadrature rules for arbitrary point sets contained in arbitrary volumes, `ASKI` computes integration weights for integration of functions on the wavefield points over the volumetric cells of the inversion grid. The inversion grid takes care of the localization of wavefield points inside the inversion grid cells and, if requested, the transformation of cells to a hexahedral (or tetrahedral) standard cell for the computation of the integration weights. Hence, some combinations of wavefield points (i.e. forward methods), integration weight types and inversion grid types are not possible.

The preintegrated kernel values are also written to files, which may be flexibly read in for arbitrary subsets of data by the independent binary programs conducting any sensitivity analysis or an iteration step in the iterative full waveform inversion. Those tools work on the sensitivity matrix, which in the FWI is used in a linear system of equations which relates data residua δd_i to a model improvement δm . In the course of the iterative full waveform inversion, it is naturally possible by `ASKI` to gradually increase the frequency content of the inverted data and to choose smaller inversion grid cells in each iteration, i.e. increasing the spatial resolution dependent on frequency.

Contents

0	Guide	11
	Full Waveform Inversion - Classical Waveform Sensitivity Kernels	12
	Cross Borehole Example	14
	Time-Domain Sensitivity Kernels	15
	Analysis of Acquisition Geometry using Kernel Focussing	18
	Full Waveform Inversion - Focused Waveform Sensitivity Kernels	19
1	Basic Steps	21
1.1	Installing ASKI	21
1.2	Create Main Parameter File	21
1.3	Iteration Step Parameter Files	21
1.4	Create Directory Environment	21
1.5	Data in ASKI	22
1.6	Prepare Measured Data	23
1.7	Define an Inversion Grid	23
1.7.1	chunksInversionGrid	24
1.7.2	schunkInversionGrid	26
1.7.3	scartInversionGrid	29
1.7.4	ecartInversionGrid	31
1.7.5	specfem3dInversionGrid	34
1.8	Define a Starting Model	34
1.9	Export Inverted Model	35
1.10	Solving the Forward Problem	35
1.11	Choose Integration Weights	36
1.12	Create a Data and Model Space	36
1.12.1	How Data Samples are Characterized	36
1.12.2	How Model Parameters are Characterized	37
1.12.3	How to Define Data and Model Space: Choosing a Set of Data Samples and Model Parameters	37

1.13	Initiate Basic Requirements	37
1.14	Compute Spectral Waveform Sensitivity Kernels	37
1.15	Transform to Time-Domain Sensitivity Kernels	38
1.16	Plot Spectral Waveform Sensitivity Kernels	38
1.17	Plot Time Sensitivity Kernels	39
1.18	Solve Kernel System	39
2	Files	41
2.1	Parameter Files	41
2.1.1	Main Parameter File	41
2.1.2	Parameter File for Specific Iteration Step	44
2.2	Event List File	46
2.3	Station List File	47
2.4	Measured Data Files	47
2.5	Synthetic Data Files	48
2.6	Vtk Files	48
2.7	Data and Model Space File	48
2.7.1	Model Values Block	49
2.7.2	Data Samples Block	49
2.8	ecartInversionGrid Files	50
2.8.1	Nodes Coordinates Files	50
2.8.2	Cell Connectivity Files	51
2.8.3	Cell Neighbours File	51
3	Programs, Scripts and Modules	53
3.1	binary programs	53
3.1.1	combineInvertedModels	53
3.1.2	computeCorrectionSyntheticData	53
3.1.3	computeDataFromKernelSystem	54
3.1.4	computeFocussedMisfit	54
3.1.5	computeKernelCoverage	54
3.1.6	computeKernels	55
3.1.7	computeMisfit	56
3.1.8	createMeasuredData	56
3.1.9	createStartmodelKim	56
3.1.10	exportKim	58

CONTENTS	9
3.1.11 focusSpectralKernels	59
3.1.12 initBasics	59
3.1.13 investigateDataResiduals	60
3.1.14 invgrid2vtk	60
3.1.15 kdispl2vtk	61
3.1.16 kernel2vtk	61
3.1.17 kgt2vtk	62
3.1.18 krm2kim	62
3.1.19 paths2vtk	63
3.1.20 solveCglsKernelSystem	63
3.1.21 solveKernelSystem	64
3.1.22 solveParKernelSystem	66
3.1.23 spec2timeKernels	67
3.1.24 timeKernel2vtk	68
3.2 Integration Weights	69
3.2.0 Compute Average (no integration)	69
3.2.1 Scattered Data Integration	70
3.2.2 Linear (first order) Integration	72
3.2.3 External Integration Weights	73
Bibliography	75
GNU Free Documentation License	77

Chapter 0

Guide

This chapter is intended to guide you, dependent on what you want to do, through all necessary steps to achieve your goals.

If you don't know about `ASKI` yet, we recommend you to quickly read through the next section, which explains some basic terminology in `ASKI` and the concepts it is based on.

The sections below address possible operations you can conduct with `ASKI`. For every operation, we only refer to the necessary basic steps (by \rightarrow), which are described in chapter 1 (page 21).

Make sure to have a complete read through before hastily doing anything!

Good Luck!

Full Waveform Inversion using pre-integrated spectral Waveform Sensitivity Kernels

Iterative inversion scheme which uses waveform sensitivity kernels to gain model updates from data residuals.

In addition to `ASKI` → 1.1 (page 21), you will need to install a supported software which solves the forward problem → 1.10 (page 35).

Before The First Iteration Step

- Create a main parameter file (e.g. in the parent directory of your specific inversion directory, or where you collect main parameter files for all your inversion projects) → 1.2 (page 21). You will need this file as an input argument to almost all programs/scripts. Set `MAIN_PATH_INVERSION` to a correct value. The directory does not need to exist yet, if not, then it will be created.
You should keep the default values of `ITERATION_STEP_PATH` and `PARFILE_ITERATION_STEP` unless you know what you are doing. If you wish to use different values for the complete inversion process, it makes sense to adjust them now.
All other parameters can be adjusted later.
- Create a directory structure for the expected number of iteration steps of your inversion → 1.4 (page 21)
- Make yourself familiar with the form of data used in `ASKI` → 1.5 (page 22).
Set `APPLY_EVENT_FILTER`, `APPLY_STATION_FILTER`, `PATH_MEASURED_DATA`, `PATH_EVENT_FILTER`, `PATH_STATION_FILTER`, `FILE_EVENT_LIST`, and `FILE_STATION_LIST`, as well as `MEASURED_DATA_FREQUENCY_STEP`, `MEASURED_DATA_NUMBER_OF_FREQ` and `MEASURED_DATA_INDEX_OF_FREQ` in your main parameter file, before preparing your data in the required form → 1.6 (page 23).
- Set `FORWARD_METHOD` in your main parameter file to the value of your choice. If you want to use different methods in the course of inverting one dataset (e.g. starting with a 1D method, continuing with a 3D method), then it may make sense to create a different directory structure for each method and using the final model of one method as the starting model for the next method.
- Choose a model parametrization by setting `MODEL_PARAMETRIZATION` in the main parameter file to a value of your choice (which is supported by your forward method).

Before Each Iteration Step (including the first one)

- Set `CURRENT_ITERATION_STEP` in your main parameter file to the correct value.
When continuing your inversion with a different method, you may also keep the current iteration step index (in order for you not to get confused) and leave subdirectories of your `MAIN_PATH_INVERSION` empty (or delete them after creation if they are not needed): e.g. an inversion with one method could start with iteration step 4 (and respective subdirectory), if you have already conducted 3 iteration steps with other methods.
- Define the inversion grid of the current iteration → 1.7 (page 23).

- Set all parameters in the specific iteration step parameter file to correct values → 1.3 (page 21), including the correct reference to your inversion grid. Refer to the documentation of your forward method on how to set filenames `FILE_WAVEFIELD_POINTS` and `FILE_KERNEL_REFERENCE_MODEL` (as the handling of these file are method dependent).
- Dependent on your method and model parametrization, take care about communicating the current model (inverted in the previous iteration) to your forward method → 1.9 (page 35). Before the first iteration, however, you need to define some starting model → 1.8 (page 34).

Conducting An Iteration Step

- Compute forward wavefields and Green tensor components w.r.t. the current model by your method. Refer to the respective documentation of your method.
After that, you may prepare the synthetic data in the way `ASKI` expects it (see sections 1.5 (page 22) and 2.5 (page 48)). Refer to the documentation of your method on how to do it.
- Set `TYPE_INTEGRATION_WEIGHTS` in the iteration specific parameter file → 1.11 (page 36). You should keep the default values of `FILE_INTEGRATION_WEIGHTS`, unless you know what you are doing (can be any name, will be created, but is referred to, afterwards. You could compute different types of weights and store them in different files, this way).
- Initiate basic requirements for all programs and scripts → 1.13 (page 37).
- Define data and model space, where the paths are mainly important for now → 1.12 (page 36).
- Compute sensitivity kernels for your specific set of paths and your set of model parameters → 1.14 (page 37).
If desired, you may have a look at your kernels → 1.16 (page 38).
- Choose a specific data and model space. You may well play around with different subsets of data or smoothing (next step) in the course of inverting for different models → 1.12 (page 36).
- Finally compute the inverted model by solving the kernel system → 1.18 (page 39).

Cross Borehole Example

This section describes the synthetic cross borehole example presented in Florian's dissertation [FS06] (chapter 5.1) as well as in our GJI paper [SFL 8] (section 4.1). It is an example of full waveform inversion using pre-integrated spectral waveform sensitivity kernels (as described in the first recipe of this chapter Guide (page 11)).

Install ASKI → 1.1 (page 21).

Then download the example files `ASKI_inversion_cross_borehole.tar.gz` via `http://www.rub.de/aski` (566 MB). In this package, all files of the first 2 iterations (plus illustrating extras) are provided. In order to keep the download small, wavefield output is not provided for iterations 3 to 12. You should be able to reproduce the wavefields immediately for any iteration step from the provided inverted models without reproducing all preceding iterations. The pre-integrated sensitivity kernel files are only provided for iterations 1 to 3. Kernel files for the other iterations can be downloaded by interested users separately via `http://www.rub.de/aski` (package `ASKI_inversion_cross_borehole_sensitivity_kernels_iter04-iter12.tar.gz`, (907 MB)).

Most importantly, you must change the value of `MAIN_PATH_INVERSION` in file `ASKI_inversion_cross_borehole/main_parfile_cross_borehole` to `"your_base_path/"` in order for all the example directory to work. If you do not want to overwrite the existing output, alternatively you should create a separate inversion directory from scratch → 1.4 (page 21) and leave `ASKI_inversion_cross_borehole/` as a reference only.

This example was generated using `SPECFEM3D_Cartesian 3.0` for ASKI. Even if you intend to use a different forward method, you can still practice the operations of the main ASKI package and, thus, learn about how to conduct any steps in the full waveform inversion *after* the forward problem is solved. Just read the description of the contained files in `ASKI_inversion_cross_borehole/README`.

If you want to reproduce also the forward wavefields from `SPECFEM3D_Cartesian 3.0`, you need to install it, of course → 1.10 (page 35).

Time-Domain Sensitivity Kernels

This section describes, how to compute time-domain waveform sensitivity kernels for a specific set of sources receivers with respect to a certain background earth model as an operation separate of any other `ASKI` operations. The kernels in time-domain are much more intuitive to look at for human beings, than the standard frequency-domain sensitivity kernels. You may, as well, compute time-domain sensitivity kernels from the kernels produced in any iteration step of a full waveform inversion (page 12). For this purpose, apply the steps “Transforming to Time-Domain Sensitivity Kernels” (below) after you computed the spectral kernels in you iteration step, as the time-domain waveform kernels are produced by an inverse Fourier transform from the standard frequency-domain waveform sensitivity kernels on which `ASKI` is based.

Please do not get confused by the general terminology of *inversion* and *iteration*, etc. Technically you will be conducting an incomplete first iteration step of a full waveform inversion, using all the program infrastructure which is also used for a full waveform inversion.

In addition to `ASKI` → 1.1 (page 21), you will need to install a supported software which solves the forward problem → 1.10 (page 35).

Preceding Considerations

- Create a main parameter file (e.g. in the parent directory of your specific inversion directory, or where you collect main parameter files for all your inversion projects or analyses) → 1.2 (page 21). You will need this file as an input argument to almost all program-/scripts.
Set `MAIN_PATH_INVERSION` to a correct value. The directory does not need to exist yet, if not, then it will be created.
Set `ITERATION_STEP_PATH` and `PARFILE_ITERATION_STEP` to desired values, or leave the default values, if present.
All other parameters can be adjusted later.
- Create a directory structure for only one iteration step → 1.4 (page 21)
- Even if you do not have any measured data, it might still be beneficial for you to make yourself (roughly) familiar with the form of data used in `ASKI` → 1.5 (page 22).
In your main parameter file, set the following values:
 - Set `FILE_EVENT_LIST` and `FILE_STATION_LIST` to define the sources and receivers which are involved in the paths that you would like to compute the kernels for.
 - Dependent on the (length of the) time series you want to deal with, define the frequency discretization of the spectral kernels that will be produced first. This must be done by `MEASURED_DATA_FREQUENCY_STEP`, `MEASURED_DATA_NUMBER_OF_FREQ` and `MEASURED_DATA_INDEX_OF_FREQ`.

In general, for the pure kernel computation you do not need any measured data. So, here you do not need to prepare data in the `ASKI` required form.

- Set `APPLY_EVENT_FILTER`, `APPLY_STATION_FILTER` (and if required `PATH_EVENT_FILTER` and `PATH_STATION_FILTER`) in your main parameter file. In case the forward wavefields were calculated w.r.t. an impulse source time function (or the

source-time function was deconvolved), you should apply filters that taper down the amplitude spectrum before the maximum frequency used. Otherwise, the inverse Fourier transform below can create artefacts.

- Set `FORWARD_METHOD` in your main parameter file to the value of your choice.
- Choose a model parametrization by setting `MODEL_PARAMETRIZATION` in the main parameter file to a value of your choice (which is supported by your forward method).
- Set `CURRENT_ITERATION_STEP` in your main parameter file to value 1, as you are technically starting to conduct the first (and only) iteration step of a full waveform inversion.
- Define the inversion grid → 1.7 (page 23), which controls the spacial volumetric discretization (resolution) of the computed sensitivity kernels. In case of just computing (time) kernels to look at, it is not crucial to regard this resolution as the resolution of some inverted model, as no inversion will be conducted on the inversion grid.
- Set all parameters in the specific iteration step parameter file to correct values → 1.3 (page 21), including the correct reference to the inversion grid. Set `ITERATION_STEP_NUMBER_OF_FREQ` and `ITERATION_STEP_INDEX_OF_FREQ` to the *same* values as the `MEASURED_DATA_NUMBER_OF_FREQ` and `MEASURED_DATA_INDEX_OF_FREQ` in the main parameter file! Refer to the documentation of your forward method on how to set filenames `FILE_WAVEFIELD_POINTS` and `FILE_KERNEL_REFERENCE_MODEL` (as the handling of these file are method dependent).
- Dependent on your method and model parametrization, define your background model with respect to which the kernels will be computed. If you have some inverted model file, use → 1.9 (page 35). For defining a starting model, see → 1.8 (page 34).

Computing Spectral Waveform Sensitivity Kernels

- Compute forward wavefields and Green tensor components w.r.t. the current model by your method. Refer to the respective documentation of your method.
After that, you may prepare the synthetic data in the way `ASKI` expects it (see sections 1.5 (page 22) and 2.5 (page 48)). Refer to the documentation of your method on how to do it.
- Set `TYPE_INTEGRATION_WEIGHTS` in the iteration specific parameter file → 1.11 (page 36). If you intend to compute the time kernels on wavefield points only, you will technically never use the integration weights, but nevertheless they need to be correctly created (in that case use a simple type of weights like 5 or 0). You should keep the default values of `FILE_INTEGRATION_WEIGHTS`, unless you know what you are doing (can be any name, will be created, but is referred to, afterwards). You could compute different types of weights and store them in different files, this way).
- Initiate basic requirements for all programs and scripts → 1.13 (page 37).
- If you have many paths, you may define a data and model space concentrating on defining paths → 1.12 (page 36). If you have only one path or just a few, it is possible (and probably also convenient) to just continue to the computation of the kernels.

- Compute the standard frequency-domain sensitivity kernels for your specific set of paths (or the one or few paths, one after another). In case you want to transform to time-domain kernels *on wavefield points* (and not pre-integrated kernels on inversion grid), use option `-wp` → 1.14 (page 37)
If desired, you may have a look at the standard frequency-domain kernels → 1.16 (page 38).

Transforming to Time-Domain Sensitivity Kernels

- Transform the standard frequency-domain waveform kernels to time domain. In case you want to get time-domain kernels *on wavefield points*, (and not pre-integrated kernels on inversion grid), you must use option `-wp` (in this case, you already should have used `-wp` to compute spectral kernels) → 1.15 (page 38).
Make sure that the wavefield spectra have a sufficiently small amplitude spectrum at high frequencies, in order for the inverse Fourier Transform to work satisfactorily. Otherwise use filters.
- Plot the time kernels → 1.17 (page 39).

Analysis of Acquisition Geometry using Kernel Focussing

THIS SECTION IS WORK IN PROGRESS!

Short introduction as to what “Optimization of Acquisition Geometry using Kernel Focussing” actually is:

todo list:

- step 1
- step 2
- ...

Full Waveform Inversion - Focused Waveform Sensitivity Kernels

THIS SECTION IS WORK IN PROGRESS!

Short introduction as to what “Full Waveform Inversion using Focused Data for Maximum Resolution” actually is:

todo list:

- step 1
- step 2
- ...

Chapter 1

Basic Steps

In general, in this chapter we provide only basic information. For more detail on specific steps or objects, we always refer to the respective sections below in this document.

1.1 Installing ASKI

- Download main package `ASKI_1.0.tar.gz` from `http://www.rub.de/aski`
- Unpack tar ball (containing the directory `ASKI_1.0/`) somewhere
- Follow the directions in file `ASKI_1.0/README`

1.2 Create Main Parameter File

The simplest way to create a specific main parameter file for your operation is to modify / adjust a copy of the template file `template/main_parfile_template`.

Refer to the commented documentation in `main_parfile_template` or to sections 2.1 (page 41) and 2.1.1 (page 41).

1.3 Iteration Step Parameter Files

Having created a directory environment for your operation, as described in section 1.4 (page 21), there should automatically have been created template parameter files in each directory of an iteration step, having filenames as defined by parameter `PARFILE_ITERATION_STEP` in the main parameter file.

Refer to the commented documentation in those template files or to sections 2.1 (page 41) and 2.1.2 (page 44).

1.4 Create Directory Environment

Call python script `create_ASKI_dir.py`

USAGE: please give 2 arguments:
 [1] main parameter file of inversion
 [2] number of iteration steps

EXAMPLE:

```
create_ASKI_dir.py ./main_parfile_Aegean1 10
```

Put your main parameter file (see 1.2 (page 21)) as the first, and the expected number of iteration steps as the second argument.

You can always *recall this script at any later time* with a larger number of iteration steps. All existing directories *will not be affected*, only additional non-existing objects will be created. Recalling this script with a smaller number of steps will not delete anything.

1.5 Data in ASKI

One certain data sample in ASKI is characterized by a seismic *source*, a *component* of a seismic *receiver*, and a *frequency*, as well as if it is *real* or *imaginary* part of the complex spectral values. It has the value of displacement of the ground in the unit of meters.

Events and Receivers

The events file (2.2 (page 46)) and stations file (2.3 (page 47)) constitute a collection of *all* events (stations) which will be involved *in any way* in your ASKI operation.

All programs/scripts will refer to a specific event (station) by its event-ID (station-name).

Receiver Components

All programs/scripts will refer to a specific receiver component by the following abbreviatory names.

Dependent on the coordinate system in which the receivers are defined (Cartesian or spherical, which is defined by the first line of the station list file), the supported names of receiver components may have a different meaning:

Cartesian receivers

CX: Cartesian X-coordinate (first Cartesian coordinate)

CY: Cartesian Y-coordinate (second Cartesian coordinate)

CZ: Cartesian Z-coordinate (third Cartesian coordinate)

N: same as $-CX$

S: same as CX

E: same as CY

W: same as $-CY$

UP: same as CZ

DOWN: same as -CZ

Spherical receivers

CX: Cartesian X-coordinate with X-axis through equator and 0°-meridian

CY: Cartesian Y-coordinate with Y-axis through equator and 90°E-meridian

CZ: Cartesian Z-coordinate with Z-axis through north pole

N: local north

S: local south

E: local east

W: local west

UP: local up

DOWN: local down

Frequency Discretization

In `ASKI`, frequencies are given by a frequency step Δf [Hz] and by a set of integer valued frequency indices.

For specific frequency index $i \in \mathbb{N}$, the corresponding frequency f_i [Hz] computes as $f_i = i \cdot \Delta f$. E.g. $\Delta f = 10$ Hz and frequency indices $i = 2, 3, 5, 7, 10$ define the set of discrete frequencies $f_i = 20.0, 30.0, 50.0, 70.0, 100.0$ [Hz].

1.6 Prepare Measured Data

For measured data given in some basic data formats like Seismic Unix and time-series given as textfiles per trace, the binary program `createMeasuredData` → 3.1.8 (page 56) converts the time-domain data to the special frequency-domain form required by `ASKI`. Call `createMeasuredData` (without arguments), then a help message will be printed how to use it.

Otherwise you might well prepare measured data files on your own as required by `ASKI`, see sections 1.5 (page 22) and 2.4 (page 47).

1.7 Define an Inversion Grid

There are different types of `ASKI` inversion grids suitable for different geometries, forward methods, hence, applications.

All inversion grids are defined by setting parameters `TYPE_INVERSION_GRID` and `PARFILE_INVERSION_GRID` in the parameter file of the current iteration step.

In the following, we present the supported inversion grid types and explain the particular parameters in the respective inversion grid parameter file.

1.7.1 `chunksInversionGrid`

The chunks inversion grid consists of 1, 2, 3 or 6 chunks of a cubed sphere (such as the schunk inversion grid). The chunk width is always 90 degrees, except for 1-chunk grids where smaller chunks are allowed. The inversion grid base cells are equi-angularly distributed, resulting in cells which are more or less evenly distributed in size (by contrast to the schunk inversion grid, where the lateral projections of the cells onto the tangential plane have an equi-distant distribution on the plane).

2 chunk grids consist of two neighbouring 90-degree chunks, 3 chunk grids of three chunks which are ALL neighbours of each other (i.e. essentially half of the Earth).

Inside the chunks, the inversion grid cells are constructed on the basis of regularly distributed base cells in the fashion of the schunk inversion grid, having a certain number of refinement blocks in depth inside which a fixed regular lateral resolution and depth resolution can be chosen. (THE FOLLOWING FEATURE IS NOT YET IMPLEMENTED: thereafter, by certain mechanisms, the base cells can be locally refined, e.g. due to ray coverage of the data)

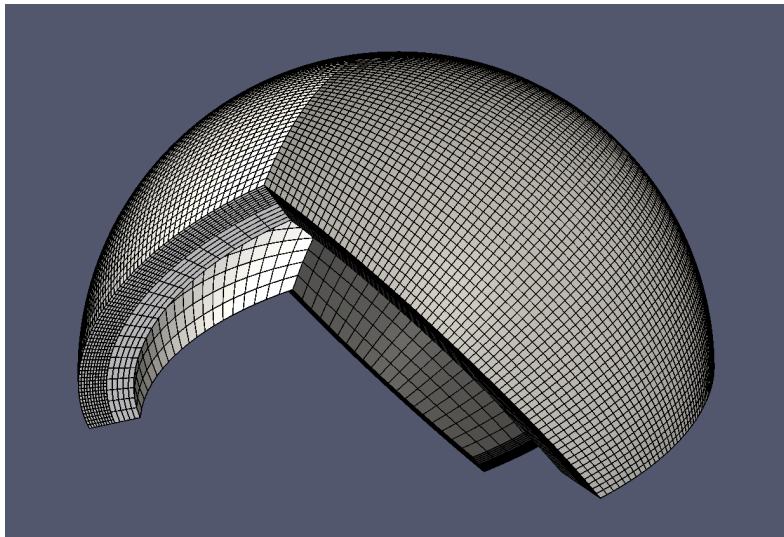


Figure 1.1: Example of a chunks inversion grid (with 3 chunks in global projection)

The inversion grid is defined via a parameter file, a template of which is file `template/chunksInversionGrid_parfile_template` (the template file defines the inversion grid shown in the above figure). The meaning of the keywords in the parameter file defining an inversion grid of type `chunksInversionGrid` are as follows:

`CHUNKS_INVGRID_GEOM_NCHUNK`

Number of chunks (1, 2, 3, or 6).

`CHUNKS_INVGRID_GEOM_RMAX`

Maximum radius (upper boundary of the chunk(s)) [must be the same unit in which wavefield points are given, usually be in km]

CHUNKS_INVGRID_GEOM_CLAT, CHUNKS_INVGRID_GEOM_CLON

Center of cubed-sphere chunk in degrees.

CHUNKS_INVGRID_GEOM_WLAT, CHUNKS_INVGRID_GEOM_WLON

Assuming $ROT = 0$.

Width of cubed-sphere chunk in degrees (must be 90.0 for $NCHUNK > 1$).

CHUNKS_INVGRID_GEOM_ROT

Rotation of chunk around local vertical counterclockwise in degrees.

CHUNKS_INVGRID_BASE_NREF_BLOCKS

Number of blocks of layers within which lateral cell width remains constant.

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

CHUNKS_INVGRID_BASE_NLAY

Number of layers within each block (integer array).

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

CHUNKS_INVGRID_BASE_THICKNESS

Thickness of layers in each block in km (real array).

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

CHUNKS_INVGRID_BASE_NLAT, CHUNKS_INVGRID_BASE_NLON

Number of cells in latitude/longitude direction for each layer block.

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

CHUNKS_INVGRID_FILE

Filename of binary inversion grid file (relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/`) to which a newly created inversion grid will be written and from which the inversion grid will be read if it exists and if not recreating the inversion grid.

VTK_PROJECTION

`VTK_PROJECTION` is one of:

'GLOBAL': center of chunk(s) in CLAT, CLON, with applied CHUNKS_INVGRID_GEOM_ROT

'LOCAL_CURV': center of chunk in $x=y=0$, NO CHUNKS_INVGRID_GEOM_ROT applied, but with curvature

'LOCAL_FLAT': center of chunk in $x=y=0$, NO CHUNKS_INVGRID_GEOM_ROT, NO curvature, i.e. projection of each depth onto the tangential x-y-plane

'LOCAL_NORTH_CURV': same as 'LOCAL_CURV', but WITH CHUNKS_INVGRID_GEOM_ROT applied, i.e. x really points to south

'LOCAL_NORTH_FLAT': same as 'LOCAL_FLAT', but WITH CHUNKS_INVGRID_GEOM_ROT applied, i.e. x really points to south

VTK_GEOMETRY_TYPE

Select the geometry type. VTK_GEOMETRY_TYPE is one of:

'CELLS': data on inversion grids will be written on the volumetric inversion grid CELLS (hexahedral) to vtk files (as UNSTRUCTURED_GRID datasets) -> intuitive volumetric view

'CELL_CENTERS': data on inversion grids will be written on the cell center POINTS to vtk files (as POLYDATA datasets) -> smaller files, better to apply filters in ParaView

SCALE_VTK_COORDS, VTK_COORDS_SCALING_FACTOR

Scale vtk geometry coordinates by factor VTK_COORDS_SCALING_FACTOR, if SCALE_VTK_COORDS = .True. this may be helpful if coordinate values (e.g. in m) get so large that they cause problems in paraview.

1.7.2 schunkInversionGrid

The spherical chunk inversion grid is derived from a simple planar cartesian grid by projection of each grid plane onto a spherical shell, assuming that the cartesian grid plane touches the spherical shell at its center. Since cells have uniform size in the planar cartesian grid plane, cells on the spherical shell become successively smaller with distance from the center of the spherical chunk. Radially, the inversion grid may consist of several layer blocks each of which may have its own cell size.

To describe points in the spherical chunk, we either use LOCAL cartesian coordinates with the center of the chunk at $x=y=0$ and $z=r$, or GLOBAL cartesian coordinates with z pointing towards the north pole, x pointing towards the equator at $\text{lon}=0$ and y pointing towards the equator at $\text{lon}=90$.

The center of the chunk may be shifted to any place on the sphere and, in addition, may be rotated counterclockwise around its local vertical.

The inversion grid is defined via a paramter file, a template of which is file `template/schunkInversionGrid_parfile_template` (the template file defines the inversion grid shown in the above figure). The meaning of the keywords in the parameter file defining an inversion grid of type `schunkInversionGrid` are as follows:

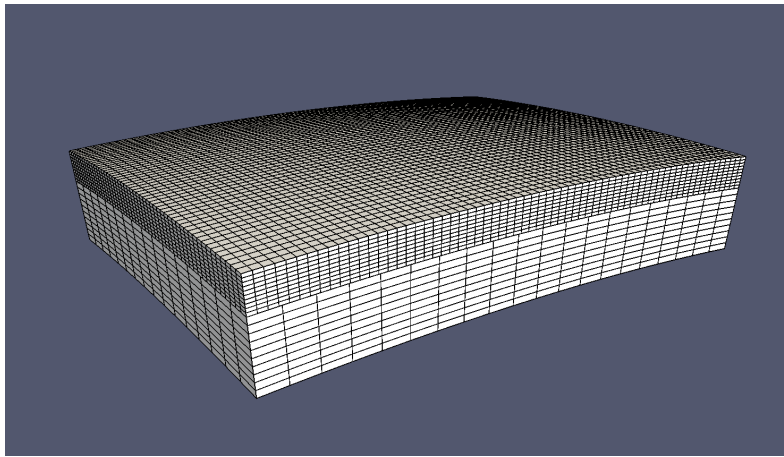


Figure 1.2: Example of a simple chunk inversion grid (global projection, note the slight curvature of the small chunk)

SCHUNK_INVGRID_CLAT, SCHUNK_INVGRID_CLON

Geographic latitude/longitude of center of spherical chunk in degrees.

SCHUNK_INVGRID_RMAX

Maximum radius (upper boundary of spherical chunk) in km.

SCHUNK_INVGRID_WLAT, SCHUNK_INVGRID_WLON

Assuming $ROT = 0$.

Width of spherical chunk parallel to longitude/latitude in degrees.

SCHUNK_INVGRID_ROT

Rotation of chunk around local vertical counterclockwise in degrees.

SCHUNK_INVGRID_NREF_BLOCKS

Number of blocks of layers within which lateral cell width remains constant.

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

SCHUNK_INVGRID_NLAY

Number of layers within each block (integer array).

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

SCHUNK_INVGRID_THICKNESS

Thickness of layers in each block in km (real array).

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

SCHUNK_INVGRID_NLAT, SCHUNK_INVGRID_NLON

Number of cells in latitude/longitude direction for each layer block.

(compare `scartInversionGrid` 1.7.3 (page 29): definition and usage of `NREF_BLOCKS`, `NLAY`, `THICKNESS`, `NX`, `NY`)

VTK_PROJECTION

`VTK_PROJECTION` is one of:

'GLOBAL': center of chunk in `CLAT`, `CLON`, with applied `SCHUNK_INVGRID_ROT`

'LOCAL_CURV': center of chunk in `x=y=0`, NO `SCHUNK_INVGRID_ROT` applied, but with curvature

'LOCAL_FLAT': center of chunk in `x=y=0`, NO `SCHUNK_INVGRID_ROT` applied, NO curvature, i.e. projection of each depth onto the tangential x-y-plane

'LOCAL_NORTH_CURV': same as 'LOCAL_CURV', but WITH `SCHUNK_INVGRID_ROT` applied, i.e. x really points to south

'LOCAL_NORTH_FLAT': same as 'LOCAL_FLAT', but WITH `SCHUNK_INVGRID_ROT` applied, i.e. x really points to south

VTK_GEOMETRY_TYPE

Select the geometry type. `VTK_GEOMETRY_TYPE` is one of:

'CELLS': data on inversion grids will be written on the volumetric inversion grid `CELLS` (hexahedral) to vtk files (as `UNSTRUCTURED_GRID` datasets) -> intuitive volumetric view

'CELL_CENTERS': data on inversion grids will be written on the cell center `POINTS` to vtk files (as `POLYDATA` datasets) -> smaller files, better to apply filters in ParaView

SCALE_VTK_COORDS, VTK_COORDS_SCALING_FACTOR

Scale all VTK point coordinates by an additional factor, if `SCALE_VTK_COORDS = .false.`, then `VTK_COORDS_SCALING_FACTOR` is ignored.

1.7.3 `scartInversionGrid`

A Simple CARTesian inversion grid covers a Cartesian cuboid which can be shifted to a certain location in Cartesian space and may be rotated about the local vertical axis. Its cells are distributed in layers. Each layer has a certain thickness and a regularly distributed number of inversion grid cells along each lateral direction of the cuboid.

Please consult the documentation of your forward method (1.10 (page 35)) if it supports inversion grids of type `scartInversionGrid`.

All coordinates, e.g. of events and stations or wavefield points, are interpreted by this type of inversion grid as X (first coordinate), Y (second coordinate), Z (third coordinate). Their units (e.g. meters or kilometers) are not assumed by the inversion grid and are essentially defined by the wavefield points, hence, they might be method dependent and must be overall consistent. Every type of integration weights is supported by this type of inversion grid, except weights of type 6 (external integration weights).

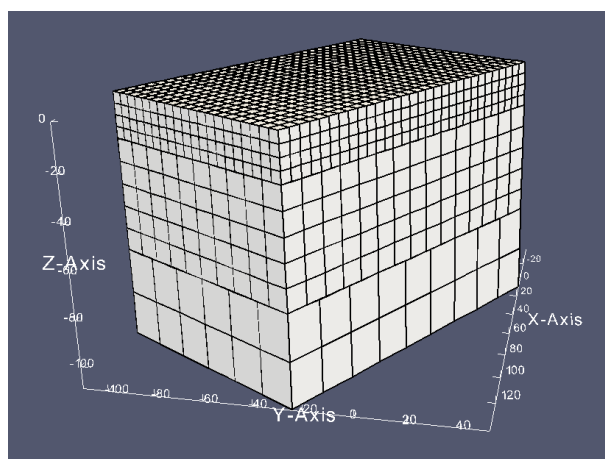


Figure 1.3: Example of a simple Cartesian inversion grid

The shape of the cuboid, as well as the distribution of inversion grid cells, are defined via a parameter file, a template of which is file `template/scartInversionGrid_parfile_template`. In the following, the particular parameters are explained, with the example values always referring to the inversion grid as displayed in figure 1.3 (page 29).

SCART_INVGRID_CX

X-coordinate of center of cuboid (real number)

Example:

```
SCART_INVGRID_CX = 50.0
```

SCART_INVGRID_CY

Y-coordinate of center of cuboid (real number)

Example:

```
SCART_INVGRID_CY = -30.0
```

SCART_INVGRID_ZMAX

Maximum Z-coordinate of cuboid (real number), i.e. Z-coordinate of the “surface” of the inversion grid

Example:

```
SCART_INVGRID_ZMAX = 0.0
```

SCART_INVGRID_WX

Width of cuboid in X-direction (real number)

Example:

```
SCART_INVGRID_WX = 100.0
```

SCART_INVGRID_WY

Width of cuboid in Y-direction (real number)

Example:

```
SCART_INVGRID_WY = 150.0
```

SCART_INVGRID_ROT

Angle in degrees of anti-clockwise rotation about the local Z-axis through the lateral center of the cuboid (real number)

Example:

```
SCART_INVGRID_ROT = 60.0
```

SCART_INVGRID_NREF_BLOCKS, SCART_INVGRID_NLAY, SCART_INVGRID_THICKNESS

For an arbitrary number of SCART_INVGRID_NREF_BLOCKS blocks of layers, the vectors SCART_INVGRID_NLAY (integer values) and SCART_INVGRID_THICKNESS (real values), both of length SCART_INVGRID_NREF_BLOCKS, define the Z-direction refinement of each block, whereby SCART_INVGRID_NLAY(i) defines the number of layers in block i, and SCART_INVGRID_THICKNESS(i) defines the thickness of all layers contained in block i.

Hence, the overall Z-direction coverage of the inversion grid is defined by SCART_INVGRID_ZMAX (which is the coordinate of the top of the first layer in the first refinement block) and SCART_INVGRID_ZMAX - SUM_i (THICKNESS(i) * NLAY(i)) (coordinate of the bottom of the last layer in last refinement block).

Example:

```
SCART_INVGRID_NREF_BLOCKS = 3
SCART_INVGRID_NLAY = 4 5 2
SCART_INVGRID_THICKNESS = 5.0 10.0 20.0
```

SCART_INVGRID_NX

Vector (of length SCART_INVGRID_NREF_BLOCKS) of integer values, defining number of inversion grid cells in X-direction, one value for each refinement block

Example:

```
SCART_INVGRID_NX = 20 10 6
```

SCART_INVGRID_NY

Vector (of length SCART_INVGRID_NREF_BLOCKS) of integer values, defining number of inversion grid cells in Y-direction, one value for each refinement block

Example:

```
SCART_INVGRID_NX = 30 15 9
```

USE_LOCAL_INVGRID_COORDS_FOR_VTK

Logical value to indicate whether to use local inversion grid coordinates for vtk geometry, i.e. no rotation by SCART_INVGRID_ROT and no shift by SCART_INVGRID_CX, SCART_INVGRID_CY, SCART_INVGRID_ZMAX (cuboid centered in X=Y=0 and ZMAX=0)

Example:

```
USE_LOCAL_INVGRID_COORDS_FOR_VTK = .false.
```

VTK_GEOMETRY_TYPE

Select the geometry type. VTK_GEOMETRY_TYPE is one of:

'CELLS': data on inversion grids will be written on the volumetric inversion grid CELLS (hexahedral) to vtk files (as UNSTRUCTURED_GRID datasets) -> intuitive volumetric view

'CELL_CENTERS': data on inversion grids will be written on the cell center POINTS to vtk files (as POLYDATA datasets) -> smaller files, better to apply filters in ParaView

SCALE_VTK_COORDS, VTK_COORDS_SCALING_FACTOR

Scale vtk geometry coordinates by factor VTK_COORDS_SCALING_FACTOR (real number), if SCALE_VTK_COORDS = .true.. This may be helpful if coordinate values (e.g. in meters) get so large that they cause problems when plotting in paraview.

Example:

```
SCALE_VTK_COORDS = .false.
VTK_COORDS_SCALING_FACTOR = 1.0
```

1.7.4 ecartInversionGrid

WARNING, EXPERIMENTAL FEATURE: *so far, this type of inversion grid works for tetrahedral cells only, since support for hexahedral cells is not completed throughout. However, even for tetrahedra the automatic detection of neighbours did not work properly in some test cases! So if you intend to use this inversion grid, please have a look at the grid and all neighbours (neighbours only required for model smoothing in case of solving the Kernel system of equations). You can do that by using binary program invgrid2vtk along with option -nb. Call invgrid2vtk (without arguments) → 3.1.14 (page 60) for further details on how to use it.*

An External CARTesian inversion grid is defined by several text files containing the definition of nodes (i.e. essentially the corner points, or rather the control nodes of the inversion grid cells) and the definition of cells by referring to the nodes. At the moment, 4-node tetrahedral cells are fully supported, and 8-node hexahedral cells are partly supported. Those files may be produced by any meshing tool. In case you are interested to export meshes your own way, section 2.8 (page 50) defines the required file formats.

ASKI provides the python module `cubit2ASKIecartInversionGrid.py` which can be used with the meshing software Cubit in a python script by first importing the module:

```
import cubit2ASKIecartInversionGrid
```

and at the very end of your meshing process calling:

```
cubit.cmd('compress all')
```

```
cubit2ASKIecartInversionGrid.export2ASKI('EXPORT_PATH')
```

whereby you may replace `EXPORT_PATH` by some location where the output files will be written.

Please consult the documentation of your forward method (1.10 (page 35)) if it supports inversion grids of type `ecartInversionGrid`.

All coordinates, e.g. of events and stations or wavefield points, are interpreted by this type of inversion grid as X (first coordinate), Y (second coordinate), Z (third coordinate). Their units (e.g. meters or kilometers) are not assumed by the inversion grid and are essentially defined by the wavefield points, hence, they might be method dependent and must be overall consistent.

Every type of integration weights is supported by this type of inversion grid, except weights of type 6 (external integration weights).

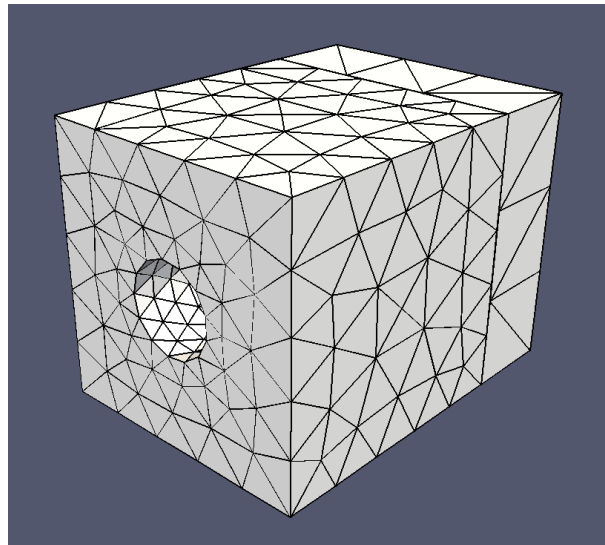


Figure 1.4: Example of an external Cartesian inversion grid created by Cubit

The nodes and cell files, e.g. produced by Cubit, are referred to in a parameter file, a template of which is file `template/ecartInversionGrid_parfile_template`. In the following, the particular parameters are explained. An example inversion grid of this type is displayed in figure 1.4 (page 32).

ECART_INVGRID_USE_NODES_COMMON

Logical value to indicate whether to use one common nodes coordinates file for all cell types (only use parameter `ECART_INVGRID_FILE_NODES` below), or to use an individual nodes coordinates file for each cell type (use parameters files `ECART_INVGRID_FILE_NODES_`

TET4, ECART_INVGRID_FILE_NODES_HEX8, ... below).

When using module `cubit2ASKIecartInversionGrid` you should set:

```
ECART_INVGRID_USE_NODES_COMMON = .True.
```

ECART_INVGRID_FILE_NODES_COMMON

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of nodes coordinates file to be commonly used for definition of cells of all types in case of `ECART_INVGRID_USE_NODES_COMMON = .True.`

When using module `cubit2ASKIecartInversionGrid` you should set:

```
ECART_INVGRID_FILE_NODES_COMMON = node_coordinates
```

ECART_INVGRID_FILE_NODES_TET4

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of nodes coordinates file to be used for definition of tet4-type cells in case of `ECART_INVGRID_USE_NODES_COMMON = .False.`

ECART_INVGRID_FILE_NODES_HEX8

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of nodes coordinates file to be used for definition of hex8-type cells in case of `ECART_INVGRID_USE_NODES_COMMON = .False.`

ECART_INVGRID_FILE_CELLS_TET4

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of cell connectivity file for definition of tet4-type cells.

When using module `cubit2ASKIecartInversionGrid` you should set:

```
ECART_INVGRID_FILE_CELLS_TET4 = cell_connectivity_tet4
```

ECART_INVGRID_FILE_CELLS_HEX8

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of cell connectivity file for definition of hex8-type cells.

When using module `cubit2ASKIecartInversionGrid` you should set:

```
ECART_INVGRID_FILE_CELLS_HEX8 = cell_connectivity_hex8
```

ECART_INVGRID_FILE_NEIGHBOURS

File name relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH/` of cell neighbours file. If not present, this file will be created when first using the inversion grid. If present, its content defines the neighbour structure of the inversion grid cells. If, however, the inversion grid is to be recreated (e.g. when calling `initBasics -recr`, see section 1.13 (page 37)), this file is recreated.

ECART_INVGRID_FILE_NEIGHBOURS_IS_BINARY

Logical value to indicate whether ECART_INVGRID_FILE_NEIGHBOURS should be binary or not.

VTK_GEOMETRY_TYPE

Select the geometry type. VTK_GEOMETRY_TYPE is one of:

'CELLS': data on inversion grids will be written on the volumetric inversion grid CELLS (hexahedral) to vtk files (as UNSTRUCTURED_GRID datasets) -> intuitive volumetric view

'CELL_CENTERS': data on inversion grids will be written on the cell center POINTS to vtk files (as POLYDATA datasets) -> smaller files, better to apply filters in ParaView

SCALE_VTK_COORDS, VTK_COORDS_SCALING_FACTOR

Scale vtk geometry coordinates by factor VTK_COORDS_SCALING_FACTOR (real number), if SCALE_VTK_COORDS = .true.. This may be helpful if coordinate values (e.g. in meters) get so large that they cause problems when plotting in paraview.

Example:

```
SCALE_VTK_COORDS = .false.
VTK_COORDS_SCALING_FACTOR = 1.0
```

1.7.5 specfem3dInversionGrid

An inversion grid of type `specfem3dInversionGrid` is method dependent and is to be used with `METHOD = SPECFEM3D` only. Whole spectral elements are used as inversion grid cells and all GLL points inside such an element as the wavefield points. All information regarding the element geometry, including information on neighbour cells and the values of the jacobian for every wavefield point contained in an element are read from files which are produced by `SPECFEM3D` methods.

Every type of integration weights is supported by this type of inversion grid, including weights of type 6 (external integration weights, i.e. the very integration weights that already `SPECFEM3D` uses to integrate over spectral elements).

Please refer to the documentation of your `SPECFEM3D` forward method (1.10 (page 35)) on how to define an inversion grid of type `specfem3dInversionGrid`.

1.8 Define a Starting Model

There are two possibilities to define an earth model for the forward wave propagation in your first iteration:

On the one hand you may use any (standard) earth model provided by the forward method you are using, if appropriate.

If this is not possible, or the models provided do not meet your needs, you may use binary `createStartmodelKim` along with the inversion grid of your first iteration (which you should have already defined) to produce an inverted model file containing some simple model on this inversion grid. Executing `createStartmodelKim` (without arguments) will print a help message how to use the program → 3.1.9 (page 56). Afterwards you may export the produced model file to your forward method as explained in section 1.9 (page 35). Template files of starting model descriptions may be found in `template/`.

1.9 Export Inverted Model

The executable `exportKim` exports an inverted model file (“kim” stands for “K”ernel “I”nverted “M”odel) along with the respective inversion grid specifications to a text file, which may be used to communicate such a model to a forward method or postprocess the model values in any way. Executing `exportKim` (without argument) will print a help message how to use it → 3.1.10 (page 58).

1.10 Solving the Forward Problem

In the following, all wave propagation codes which are supported by `ASKI` are listed. Refer to the given documentation on any details regarding the interaction of the forward codes with `ASKI`.

Gemini II

`Gemini II` is supported in this release version, for Cartesian as well as spherical setting. The `Gemini` routines producing spectral output for `ASKI`, however, might not yet be available. We hope to be able to provide them in due time. Until now, please contact us via <http://www.rub.de/aski> if you want to use `Gemini II` as a forward solver for `ASKI`.

SPECFEM3D_Cartesian

The Cartesian spectral element code `SPECFEM3D_Cartesian`, version 3.0 is supported by `ASKI`, cf. [Sch15].

SPECFEM3D_GLOBE

The global spectral element code `SPECFEM3D_GLOBE`, version 7.0, is fully supported in this release version. However, the extension package for `SPECFEM3D_GLOBE` will not be available until the beginning of 2016.

NEXD

Support for the nodal discontinuous Galerkin code `NEXD` by Lasse Lambrecht (Ruhr-University Bochum, Germany) is being implemented and tested at the moment. `NEXD`, including `ASKI`

support, will be available under terms of the GPL soon.

1.11 Choose Integration Weights

In order to numerically integrate the sensitivity kernels, which are computed on the wavefield points, over the inversion grid cells by a weighted summation of values, there are different types of integration weights provided, following different rules of integration.

The integer values of the type have the following meaning:

- 0 → all weights are the same, $\text{weight} = 1/\text{number_of_points_in_box}$, i.e. no integration(!), just building the average sensitivity value (e.g. convenient for comparison of sensitivities computed with different methods on different forward grids)
- 1 → Scattered Data Integration, as in [Lev99], polynomial degree 1
- 2 → Scattered Data Integration, as in [Lev99], polynomial degree 2, i.e. approximation order 3 (?)
- 3 → Scattered Data Integration, as in [Lev99], polynomial degree 3, i.e. approximation order 4 (?)
- 4 → for each cell, compute the highest possible order of Scattered Data Integration integration after [Lev99] (trying types 3,2,1 (in that order) until computation was successful)
- 5 → average of function values, multiplied with volume of box (i.e. \sim linear integration)
- 6 → external integration weights, to be used along with a suitable inversion grid (e.g. of type `specfem3dInversionGrid`, see section 1.7.5 (page 34))

A detailed description of some of the integration weights, especially the weights after [Lev99] can be found in section 3.2 (page 69).

1.12 Create a Data and Model Space

In order to choose a set of data samples which to invert and a set of model parameters which to invert for, you need to define a data space and a model space. Essentially, if you have m data samples, the space in which the data live is just \mathbb{R}^m (analogously, for n model parameters, the model lives in \mathbb{R}^n). You only need to define which data sample (model parameter) refers to which dimension (i.e. entry in vector) of the data space (model space).

The $m \times n$ sensitivity kernel matrix will then connect a vector of model updates from model space in \mathbb{R}^n to your specific data vector from \mathbb{R}^m .

In the following, we describe how data samples and model parameters are characterized in this software package and how you can choose specific subsets to be used.

1.12.1 How Data Samples are Characterized

As the sensitivities are calculated in frequency domain, the data live in frequency domain, too. A data sample is uniquely characterized by a seismic *source*, a *component* of a seismic *receiver*,

and a *frequency*, as well as whether it is the *real* or *imaginary* part of the complex spectral values. Refer to 1.5 (page 22) for details on data in `ASKI`.

1.12.2 How Model Parameters are Characterized

A model parameter is uniquely characterized by a parameter name (must be a valid parameter name of the model parametrization as defined by `MODEL_PARAMETRIZATION 2.1.1` (page 42)) and an inversion grid cell index in valid range.

1.12.3 How to Define Data and Model Space: Choosing a Set of Data Samples and Model Parameters

Create a text file as described in section 2.7 (page 48), e.g. by adjusting the provided template files in `template/`.

1.13 Initiate Basic Requirements

Use the executable `initBasics`. Executing `initBasics` (without arguments) will print a help message how to use it → 3.1.12 (page 59).

It first checks if all parameters needed are present in the parameter files and then creates all basic requirements for `ASKI` operations:

It reads in required files like event list and station list files, the wavefield points and the kernel reference model.

Furthermore, it creates the inversion grid (possibly storing some inversion grid files, dependent on the type of grid), localizes the wavefield points inside it and computes the integration weights, which are written to file. Once those files exist, `initBasics` and all other programs will always read the integration weights (and possibly (part of) the inversion grid) from file, *regardless* of what the parameter files say! So if at some you point want to use different integration weights or a different inversion grid, you will have to either delete the respective file(s) and rerun `initBasics`, or run `initBasics -recre` in order to recreate them.

Also a lot of `.vtk` files with statistics are produced having base filename `FILEBASE_BASIC_STATS` as defined in the parameter file of the current iteration step. Those files mainly regard the inversion grid, the wavefield points and the integration weights, where the respective filenames are extended (by something with `“vtk”`). It is *highly recommended* to call `initBasics -recre` in order to assure that all those `.vtk` files are produced and to actually have a look at them before continuing any `ASKI` operation!

1.14 Compute Spectral Waveform Sensitivity Kernels

The kernels are computed by combining green tensor component(s) and forward wavefield for a given event-station pair (called “path” below). By default, the kernels are integrated over the inversion grid cells. I.e. there is one sensitivity kernel file for a specific event-station path. This file contains sensitivity values of the requested receiver components for the requested model parameters of your model parametrization, with the values living on the inversion grid cells.

Alternatively, kernels can be computed on the wavefield points, without pre-integration (for inspection only, cannot be used for waveform inversion in that case).

Use executable `computeKernels`. Executing `computeKernels` (without arguments) will print a help message how to use it → 3.1.6 (page 55). It makes sense, to only compute kernel files for those event-station paths that you are going to use (defined by your data-model-space file).

You can define the set of event-station paths, receiver components and model parameters for which sensitivities should be computed in two ways:

- way 1 compute a kernel for only one event-station path, defined by eventID and station name using options `-evid`, `-stname`, `-comp`, `-param`
- way 2 input a data-model-space file (as defined by 1.12 (page 36)) by option `-dmspace`, defining all event-station paths and respective receiver components and model parameters for which kernels should be computed; optionally define range of path index by `-ipath1`, `-ipath2`

Setting option `-wp` will compute plain kernel values on wavefield points (no pre-integration onto inversion grid). *This option will produce separate files, so you can compute both, kernels on wavefield points and on inversion grid.*

1.15 Transform to Time-Domain Sensitivity Kernels

The time kernels are computed from the standard frequency-domain kernels (which were computed path-wise) by applying an inverse Fourier transform.

Use executable `spec2timeKernels`. Executing `spec2timeKernels` (without arguments) will print a help message how to use it → 3.1.23 (page 67).

The set of event-station paths, receiver components and model parameters for which time kernels should be produced are defined in the same “two ways” (by the same options) as computing spectral waveform kernels → 1.14 (page 37).

Additionally, you need to define the time discretization of the produced time kernel by options `-dt`, `-nt1`, `-nt2`, `-t0`.

1.16 Plot Spectral Waveform Sensitivity Kernels

One way to plot a specific sensitivity Kernel in frequency domain, i.e. the sensitivity spectra for a specific event-station path, is to produce vtk files using executable `kernel2vtk`. Executing `kernel2vtk` will print a help message how to use it → 3.1.16 (page 61).

Please note, that the output `.vtk` files (one for every frequency) might get large, dependent on the resolution of the inversion grid, since the geometry information of the inversion grid cells is contained in each `.vtk` file. If the files become too large for you, you might consider setting `DEFAULT_VTK_FILE_FORMAT = BINARY` in your main parameter file.

1.17 Plot Time Sensitivity Kernels

One way to plot a specific sensitivity Kernel in time domain, is to produce vtk files using executable `timeKernel2vtk`. Executing `timeKernel2vtk` (without arguments) will print a help message how to use it → 3.1.24 (page 68).

Please note, that the output `.vtk` files (one for every time step) might get large, dependent on the resolution of the inversion grid, since the geometry information of the inversion grid cells is contained in each `.vtk` file. If the files become too large for you, you might consider setting `DEFAULT_VTK_FILE_FORMAT = BINARY` in your main parameter file. For a lot of time steps, the `.vtk` file format is actually not optimally chosen by `ASKI`, since the complete geometry information is contained in every time-step file (a lot of redundant information is written to hard disc).

1.18 Solve Kernel System

At the moment, there are 3 executable programs implemented which solve the kernel linear system.

- `solveKernelSystem` is a serial program which sets up the kernel matrix, reads in synthetic and measured data, adds regularization conditions to the system if requested (smoothing, damping) and solves the system based on LAPACK libraries. Executing `solveKernelSystem` (without arguments) will print a help message how to use it → 3.1.21 (page 64).
- `solveParKernelSystem` does essentially the same as `solveKernelSystem`, but uses parallelized ScaLAPACK libraries to solve the (regularized) kernel linear system. You must compile executable `solveParKernelSystem` separately (`make all` does not compile it, make sure you have ScaLAPACK libraries installed and linked in the Makefile). Executing `solveParKernelSystem` (without arguments) will print a help message how to use it → 3.1.22 (page 66). This executable requires an additional parameter file, see template file `templated/solveParKernelSystem_parfile_template`.
- `solveCglsrKernelSystem` sets up the (regularized) kernel linear system, as the above executables do, but the uses a conjugate gradient algorithm to solve the kernel linear system in a least-squares sense → 3.1.20 (page 63). This executable requires an additional parameter file, see template file `templated/solveCglsKernelSystem_parfile_template`. Executing `solveCglsKernelSystem` (without arguments) will print a help message how to use it → 3.1.20 (page 63).

Chapter 2

Files

This chapter collects documentation on file formats involved in `ASKI`.

2.1 Parameter Files

Parameter files are simple text files.

The following type of lines are ignored:

- comment lines, i.e. lines STARTING with an arbitrary number of blanks followed by a “#” character
- empty lines and lines containing blanks only
- lines not containing any “=” character

How to specify one parameter:

- valid lines have the form “keyword = value” (blanks leading or following “keyword”, “=”, or “value” are ignored)
- in a valid line, all characters in front of “=” (without leading and appending blanks) are interpreted as the keyword, allowing for blank characters within the keyword (e.g. for lines “ key word = value ”, the string “key word” is used as the keyword)
- all characters behind “=” (without leading and appending blanks) are interpreted as the value string from which the value is read, which in particular means that “#” comments at the end of a line (such as “ keyword = value # comment ”) are *not* allowed!

By convention, specify *paths* (i.e. directory names, which will be concatenated with a filename of a file in that directory) always ending on “/” and specify *filenames* always *without* leading “/”.

2.1.1 Main Parameter File

Here, shortly all keywords required in the main parameter file for your specific program operation, are described.

FORWARD_METHOD

GEMINI

SPECFEM3D \rightarrow SPECFEM3D_Cartesian and SPECFEM3D_GLOBE

NEXD

For details on the methods and references to their documentation, refer to section 1.10

MODEL_PARAMETRIZATION

isoLameSI \rightarrow isotropic Lamé parameters and density in SI units: ρ (rho) [kg/m³], λ (lambda) [Pa], μ (mu) [Pa]

isoVelocitySI \rightarrow isotropic seismic velocities and density in SI units: ρ (rho) [kg/m³], v_p (vp) [m/s], v_s (vs) [m/s]

isoVelocity1000 \rightarrow isotropic seismic velocities and density in units involving a factor of 1000: ρ (rho) [g/cm³], v_p (vp) [km/s], v_s (vs) [km/s]

Nothing other parameterization supported yet.

MAIN_PATH_INVERSION

All subpaths for filenames are considered relative to this main path. This directory is thought to contain all your relevant output and (temporary) data.

Example: MAIN_PATH_INVERSION = /data/inversions/Aegean1/

PARAMETER_CORRELATION_MODE

Any model parameters can be correlated to others which are not inverted for. So far supported correlation modes:

NONE \rightarrow no parameter correlation will be done whatsoever

CORRELATE_KERNELS \rightarrow will add correlated kernel values of other parameters to the kernels of those parameters which are inverted for (experimental feature, not yet tested in practice).

PARAMETER_CORRELATION_FILE

File (relative to MAIN_PATH_INVERSION) containing the definition of the correlation and the correlation coefficients. The given filename is ignored in case of PARAMETER_CORRELATION_MODE = NONE.

CURRENT_ITERATION_STEP

Example: CURRENT_ITERATION_STEP = 3

ITERATION_STEP_PATH

Relative to main path, defining name of subdirectory of MAIN_PATH_INVERSION which contains all relevant (meta)data of an inversion step. A three-digit integer (= CURRENT_ITERATION_STEP) and “/” will be appended to ITERATION_STEP_PATH (i.e. “001/”, “002/”, ...) defining the first, second ... iteration step directory.

Example (default): ITERATION_STEP_PATH = iteration_step_

PARFILE_ITERATION_STEP

File name of iteration step specific parameter file, relative to MAIN_PATH_INVERSION/ITERATION_STEP_PATH Example: PARFILE_ITERATION_STEP = iter_parfile

PATH_MEASURED_DATA, APPLY_EVENT_FILTER, PATH_EVENT_FILTER, APPLY_STATION_FILTER, PATH_STATION_FILTER

Paths and flags related to the measured data. The paths are *absolute*, i.e. can be everywhere, e.g. close to where you have stored/processed your (time domain) data, or in directory MAIN_PATH_INVERSION, etc. ...

If you do not wish to use any filtering per events, you can switch to

APPLY_EVENT_FILTER = .false.,

if you do not wish to use any filtering per stations, you can switch to

APPLY_STATION_FILTER = .false..

In these cases, the respective filter files are not expected to exist and the respective filter paths are ignored.

The naming convention of the files in the respective directories is:

FILE_MEASURED_DATA: data_EVENTID_STATIONNAME_COMP,

FILE_EVENT_FILTER: filter_EVENTID,

FILE_STATION_FILTER: filter_STATIONNAME_COMP,

where station filters are dependent on component and STATIONNAME and EVENTID are defined in FILE_STATION_LIST and FILE_EVENT_LIST file, and COMP is a valid component (for valid components see 1.5)

Example:

```
PATH_MEASURED_DATA = /mydata/your_name_of_inversion/ASKI_data/
PATH_EVENT_FILTER = /mydata/your_name_of_inversion/ASKI_event_filter/
PATH_STATION_FILTER = /mydata/your_name_of_inversion/ASKI_station_
filter/
```

FILE_EVENT_LIST

Absolute filename where ASKI finds a text file defining a set of events in the required format (2.2)

Example: FILE_EVENT_LIST = /mydata/your_name_of_inversion/ASKI_events

FILE_STATION_LIST

Absolute filename where ASKI finds a text file defining a set of stations in the required format (2.3)

Example: `FILE_STATION_LIST = /mydata/your_name_of_inversion/ASKI_stations`

MEASURED_DATA_FREQUENCY_STEP, MEASURED_DATA_NUMBER_OF_FREQ, MEASURED_DATA_INDEX_OF_FREQ

Discretized frequency window of measured data (same expected in `event_filter/station_filter`!) given by a frequency step `FREQUENCY_STEP` [Hz] and a vector of frequency indices `INDEX_OF_FREQ` (of length `NUMBER_OF_FREQ`), where for specific frequency index i the corresponding frequency f_i [Hz] computes to $f_i = i \cdot \text{FREQUENCY_STEP}$

Example:

`MEASURED_FREQUENCY_STEP = 10.`

`MEASURED_NUMBER_OF_FREQ = 5`

`MEASURED_INDEX_OF_FREQ = 2 3 5 7 10`

which corresponds to the 5 frequencies 20, 30, 50, 70, 100 Hz

UNIT_FACTOR_MEASURED_DATA

Indicate the unit factor of the measured data. The definition of the factor is:

Multiplication of the measured data (displacement) with this factor gives values in the SI unit of meters [m].

Example: If you use seismic time-domain data for ASKI in the unit of nano meters [nm], then `UNIT_FACTOR_MEASURED_DATA` must be set to `1.0e-9`, because the data values in nano meters [nm] must be divided by $1.0 \cdot 10^9$ in order to get meters [m]. If you use seismic displacement data in the SI unit of meters, then `UNIT_FACTOR_MEASURED_DATA` must be set to `1.0`.

DEFAULT_VTK_FILE_FORMAT

Either `BINARY` or `ASCII` defining the default type of `vtk` files which will be produced in the course of running the programs.

2.1.2 Parameter File for Specific Iteration Step

Here, shortly all keywords required in a parameter file for a specific iteration step, i.e.

`MAIN_PATH_INVERSION/ITERATION_STEP_PATH/PARFILE_ITERATION_STEP`, are described.

These parameter files are created automatically by script `py/create_ASKI_dir.py`. Alternatively, you can look at the template file `template/iter_parfile_template`.

USE_PATH_SPECIFIC_MODELS

Set the flag to `.true.` if you intend to use the ASKI functionality of path specific kernel reference models (usually only for the very first iteration and a 1D method). If you want to use one global kernel reference model (for all paths, i.e. kernels) set this flag to `.false.` .

ITERATION_STEP_NUMBER_OF_FREQ, ITERATION_STEP_INDEX_OF_FREQ

Frequency discretization of this iteration step, must be a subset of global frequency discretization for this inversion defined as defined by 2.1.1.

`ITERATION_STEP_NUMBER_OF_FREQ` must be smaller or equal to `MEASURED_DATA_NUMBER_OF_FREQ` and vector

`ITERATION_STEP_INDEX_OF_FREQ` (of length `ITERATION_STEP_NUMBER_OF_FREQ`) must only contain indices contained in `MEASURED_DATA_INDEX_OF_FREQ`.

All indices here are assumed in accordance with the global frequency step `MEASURED_DATA_FREQUENCY_STEP`.

TYPE_INVERSION_GRID, PARFILE_INVERSION_GRID

Type of inversion grid (as supported, cf. 1.7) and corresponding filename of parameter file defining this inversion grid, relative to

`MAIN_PATH_INVERSION/ITERATION_STEP_PATH/`

TYPE_INTEGRATION_WEIGHTS

Type of integration weights (integer number), cf. 1.11 for supported values.

FILE_INTEGRATION_WEIGHTS

Filename of the integration weights file, which will be created and used, relative to `MAIN_PATH_INVERSION`.

FILE_WAVEFIELD_POINTS

Filename of the wavefield points file, relative to `MAIN_PATH_INVERSION`, which is in general created by the method you are using. Just refer here to this file.

FILE_KERNEL_REFERENCE_MODEL

Dependent on the method you are using, these filenames may be handled individually. Please refer to the respective documentation of the methods for recommendations how to use these parameters, or which naming to choose.

FILEBASE_BASIC_STATS

Base filename of vtk stats output files (related to inversion grid, wavefield points, integration weights, events, stations), relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH`.

PATH_OUTPUT_FILES

Folder relative to which some sensitivity analysis and inversion programs may write their output (relatively small output like models, coefficients etc., NO wavefields/kernels etc.), relative to `MAIN_PATH_INVERSION/ITERATION_STEP_PATH`. Make sure the path ends on “/”.

PATH_KERNEL_DISPLACEMENTS

Subdirectory of current iteration step path `MAIN_PATH_INVERSION/ITERATION_STEP_PATH` which contains the kernel displacement files. Make sure the path ends on “/”.

PATH_KERNEL_GREEN_TENSORS

Subdirectory of current iteration step path `MAIN_PATH_INVERSION/ITERATION_STEP_PATH` which contains the kernel green tensor files. Make sure the path ends on “/”.

PATH_SENSITIVITY_KERNELS

Subdirectory of current iteration step path `MAIN_PATH_INVERSION/ITERATION_STEP_PATH` which contains the velocity kernel files. Make sure the path ends on “/”.

PATH_SYNTHETIC_DATA

Subdirectory of current iteration step path `MAIN_PATH_INVERSION/ITERATION_STEP_PATH` which contains the files with synthetic data. Make sure the path ends on “/”.

PATH_KERNEL_REFERENCE_MODELS

This path and any contained files are only relevant for `USE_PATH_SPECIFIC_MODELS = .true.!` Subdirectory of current iteration step path `MAIN_PATH_INVERSION/ITERATION_STEP_PATH` in which path-specific kernel reference models are expected. The reference model files are assumed to have names like `krm_EVENTID_STATIONNAME`. Make sure the path ends on “/”.

2.2 Event List File

Please find the template event list file `template/file_event_list_template`.

- first line contains single character “C” or “S”, defining the coordinate system (“C”artesian or “S”pherical) with respect to which the given event coordinates lat,lon are interpreted

- each following non-empty line of the file is interpreted as a definition of one event and must contain the following space-separated values:

eventid 13 character name, (e.g. 2006.10.2977 or 061113_141238) should *not* contain whitespace!

origintime characters of form `yyyymmdd_hhmmss_nnnnnnnnn` or `yyyymmdd_hhmmss` (i.e. with or without nano-seconds), e.g. 20130320_170012 or 20130320_170002_718000000

lat latitude in degrees, $-90 \leq \text{lat} \leq 90$ (“S”) or first coordinate in wavefield points / inversion grid - frame (“C”) → read the section on inversion grid definitions (1.7)

lon longitude in degrees, $0 \leq \text{lon} \leq 360$ (“S”) or second coordinate in wavefield points / inversion grid -frame (“C”) (read 1.7)

depth source depth in km (“S”), or third coordinate in wavefield points / inversion grid -frame (“C”) (read 1.7)

mag factor on source mechanism

typ source type: 0 = force, 1 = moment tensor, -1: not specified

mom/free either 3 values (force vector) or 6 values (moment tensor)

2.3 Station List File

Please find the template station list file `template/file_station_list_template`.

- first line contains single character “C” or “S”, defining the coordinate system (“C”artesian or “S”pherical) with respect to which the given event coordinates lat,lon are interpreted
- each following non-empty line of the file is interpreted as a definition of one station and must contain the following space-separated values:

station_name 5 character name, which should *neither* contain whitespace *nor* underscours “_”!

network_code 6 character network code

lat latitude in degrees, $-90 \leq \text{lat} \leq 90$ (“S”) or first coordinate in wavefield points / inversion grid - frame (“C”) → read the manual on inversion grid definitions (1.7)

lon longitude in degrees, $0 \leq \text{lon} \leq 360$ (“S”) or second coordinate in wavefield points / inversion grid -frame (“C”) (read 1.7)

elevation altitude of station (“S”), or third coordinate in wavefield points / inversion grid - frame (“C”) (read 1.7)

2.4 Measured Data Files

All measured data files are expected to be in the directory `PATH_MEASURED_DATA` as defined in the main parameter file.

One measured data file contains all data values for one specific receiver component and a specific event. Its filename is by convention `data_EVENTID_STATIONNAME_COMP`

The files are text files containing 1 column of MEASURED_DATA_NUMBER_OF_FREQ complex numbers, which can be understood by FORTRAN read command, i.e. of form (real_part , imag_part) .

Line i contains the measured data value for the i^{th} frequency, as defined by vector of indices MEASURED_DATA_INDEX_OF_FREQ and frequency step MEASURED_DATA_FREQUENCY_STEP.

In particular, this means that *all* measured data files must contain the *same* frequency discretization, given by parameters MEASURED_DATA_INDEX_OF_FREQ, MEASURED_DATA_FREQUENCY_STEP of the main parameter file.

2.5 Synthetic Data Files

All synthetic data files are expected to be in the directory PATH_SYNTHETIC_DATA as defined in the parameter file of the current iteration step.

One synthetic data file contains the complete synthetic data values for one specific combination of event, station and receiver component. Its filename is by convention synthetics_EVENTID_STATIONNAME_COMP

The files are text files containing 1 column of ITERATION_STEP_NUMBER_OF_FREQ complex numbers, which can be understood by FORTRAN read command, i.e. of form (real_part , imag_part) .

Line i contains the synthetic data value for the i^{th} frequency, as defined by vector of indices ITERATION_STEP_INDEX_OF_FREQ and frequency step MEASURED_DATA_FREQUENCY_STEP.

2.6 Vtk Files

For visualization of basic objects of the inversion, such as the inversion grid, the wavefield points, the integration weights etc., as well as some inversion results and models, we use the vtk file format. The files contain *both*, geometry information and data values on geometric objects. The format can be either ASCII or BINARY, controlled by flag DEFAULT_VTK_FILE_FORMAT in main parfile.

General information on this file format may be found under www.vtk.org/VTK/img/file-formats.pdf

ASKI uses different software modules for writing vtk based on wavefield points, or inversion grids or event-station path lines (or event, station coordinates).

2.7 Data and Model Space File

Files in which a data and model space is defined have the following form. *It might be very instructive to also have a look at the provided example template files* template/data_model_space_info_template_* .

The blocks described in the subsections below should be put into a text file, one after another, in arbitrary order, i.e. it does not matter which block comes first (DATA SAMPLE or MODEL

VALUES). It is also supported to provide only one of the blocks, if the other one is not required for some application.

There can be arbitrary commentary in front of a block, between blocks and after the blocks. However, any line starting with DATA SAMPLES or MODEL VALUES is interpreted as the starting line of the respective block. Inside the blocks, there must not be unexpected lines!

2.7.1 Model Values Block

line: MODEL VALUES

this line defines that the model values block (definition of the model values) starts here.

line: INVERSION_GRID_CELLS value

where value is either ALL (all inversion grid cells are taken) or SPECIFIC (specific definition of set of invgrid cells following below)

line: PARAMETERS value

where value is either ALL (all inversion grid cells are taken) or SPECIFIC (specific definition of model parameters for each invgrid cell, following below. Only allowed if INVERSION_GRID_CELLS SPECIFIC).

If PARAMETERS ALL, line: nparam param_1 ... param_n

defines the parameters used for all inversion grid cells (assuming MODEL_PARAMETRIZATION defined in main parameter file).

If INVERSION_GRID_CELLS SPECIFIC, the following line must contain the number of cells ncell which should be taken, followed by ncell blocks of lines, each defining an inversion grid cell. In case of PARAMETERS ALL, these blocks consist of a single line line containing an inversion grid cell index. In case of PARAMETERS SPECIFIC, these blocks consist of two lines: one line containing an inversion grid cell index and an additional second line of the form

nparam param_1 ... param_n

defining the parameters to be used for this specific inversion grid cell (assuming MODEL_PARAMETRIZATION defined in main parameter file).

2.7.2 Data Samples Block

line DATA SAMPLES

this line defines that the data samples block starts (definition of data samples) here.

line of form: WEIGHTING value, where value is one of is one of NONE, BY_PATH, BY_FREQUENCY, BY_PATH_AND_FREQUENCY

In case of NONE, all data weights are internally set to 1.0 (i.e. no actual weighting is performed). Values BY_PATH and BY_PATH_AND_FREQUENCY are only allowed in case of PATH SPECIFIC, in which case the pairs evid staname in a specific path definition (below) is expected to be followed by one number > 0.0 and ≤ 1.0 (the weight). The frequency dependent weighting values (in cases BY_FREQUENCY, BY_PATH_AND_FREQUENCY) are defined in a separate line following the lines of form nfreq ifreq_1 ... ifreq_n (for either case of FREQUENCIES ALL or FREQUENCIES SPECIFIC). This separate lines have themselves the form nfreq w_1 ... w_n defining nfreq weights in range > 0.0 and ≤ 1.0 . In case BY_PATH_AND_FREQUENCY, both weights (for path and frequency) are *multiplied* for each data sample.

line of form: `PATHS value`, where `value` is either `ALL` (all paths for a given set of event and station indices are used), or `SPECIFIC` (a specific definition of paths as a series of event and station index pairs follows below)

If `PATHS ALL`, the next two lines are of form `nev iev_1 ... iev_n` and `nstat istat_1 ... istat_n`, defining the set of event and station indices, which form (by all combinations) the used paths.

line of form: `COMPONENTS value`, where `value` is either `ALL` (for all paths, the same components are used) or `SPECIFIC` (only allowed if `PATHS SPECIFIC`, for each path a specific set of components may be defined)

If `COMPONENTS ALL`, the next line is of form `ncomp comp_1 ... comp_n` defining the component indices for all paths.

line of form: `FREQUENCIES value`, where `value` is either `ALL` (for all paths, the same frequency indices are used) or `SPECIFIC` (only allowed if `PATHS SPECIFIC`, for each path a specific set of frequency indices may be defined)

If `FREQUENCIES ALL`, the next line is of form `nfreq ifreq_1 ... ifreq_n` defining the frequency indices for all paths.

line of form: `IMRE value`, where `value` is either `ALL` (for all paths, the same set of imaginary/real parts are used) or `SPECIFIC` (only allowed if `PATHS SPECIFIC`, for each path a specific set of imaginary/real parts may be defined)

If `IMRE ALL`, the next line is of form `nimre imre_1 ... imre_n` defining imaginary (i.e. `imre_i = im`) or real parts (`imre_i = re`) for all paths.

If `PATHS SPECIFIC`, the following line must contain the number `npaths` of paths which should be used, followed by `npaths` blocks of lines, each defining the path and the data samples for that path.

These blocks consist of at least one line containing the event /station index pair `iev istat`. For each keyword `COMPONENTS`, `FREQUENCIES` and `IMRE` – if `SPECIFIC` – one line is added to such a block of lines, in the same form as the line following keyword `ALL` (see above), defining the specific components, frequencies or set of imaginary/real parts for each of the specific paths.

2.8 ecartInversionGrid Files

2.8.1 Nodes Coordinates Files

These files contain a collection of points in space, given in Cartesian X-, Y-, Z-coordinates. They must be text files and have the following format.

The first line contains a single integer value, indicating the number of lines to come (i.e. the number of points).

Each following line contains 3 floating point numbers (separated by white space) defining Cartesian X-, Y-, Z-coordinates of a point.

2.8.2 Cell Connectivity Files

These files contain the definition of cells, based on points as defined in the nodes coordinates files. They must be text files and have the following format.

The first line contains a single integer value, indicating the number of lines to come (i.e. the number of cells).

Each following line contains n integer numbers (separated by white space, $n = 4$ in case of tet4-type cells, $n = 8$ in case of hex8-type cells), which define the control nodes of the cell and correspond to the point indices in the respective nodes coordinates file, whereby the lowest point index is 1, corresponding to the second line (first point) in the nodes coordinates file.

The order of the point indices in a line is assumed to correspond to the vtk cell conventions! In case one of the cell connectivity files not existing, or their first line containing value 0, no cells of the respective type will be created.

2.8.3 Cell Neighbours File

The terminology “lines” below refers to the case of this file not being binary, but a text file. In case of this file being binary, the file content is expected value by value as on the rows of the text file. It will be opened by FORTRAN code with attribute `access='stream'` (i.e. expecting the values as a simple byte stream) and expects integer values of `kind=4`.

The first line contains the total number of inversion grid cells `ncell`.

The next `ncell` lines (one for each cell in order of the cell index) are of the form:

`nnb icell_1 ... icell_nnb`

whereby `nnb` is the number of neighbours of the respective cell (must be 0 if no neighbours) followed by `nnb` cell indices `icell_1 ... icell_nnb`, defining the neighbour cells, if there are any neighbours.

Chapter 3

Programs, Scripts and Modules

This chapter collects scripts, binary programs or modular program components contained in the ASKI package. It is not referred to any code, here, but give details on application by the user or background knowledge.

3.1 binary programs

The commands are written in the form:

command [-optional arguments]...[positional arguments]

Commands with several positional arguments have to be written with all positional arguments in the right order (the order given in the subsection).

3.1.1 combineInvertedModels

Compute linear combination `coef1 * kim1 + coef2 * kim2` of two models on inversion grid `kim1, kim2`.

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

3.1.2 computeCorrectionSyntheticData

Computes additional files in iteration-step-specific subdirectory `SYNTHETIC_DATA` which are named as `corr_EVENTID_STATIONNAME_COMPONENT`. These files contain the quantities c_i^0 (as defined in our GJI paper eq. (26)), which are corrections to synthetic data due to change from path to global reference model.

positional arguments

dmsi_file Data-model-space-info file

main_parfile Main parameter file of inversion.

3.1.3 computeDataFromKernelSystem

Computes the 'measured' data d in the equation $d-s(-sc) = K (m1 - mref)$ for given model $m1$.

Can be used for (old-fashioned) linear computation of checkerboard data by forward multiplication of the sensitivity matrix with an artificial model vector containing checker anomalies.

positional arguments

dmsi_file Data-model-space-info file

kim_file kernel-inverted-model file which contains model $m1$.

outdir_data Output directory where the new 'measured' data files are written to.

main_parfile Main parameter file of inversion.

3.1.4 computeFocussedMisfit

compute focussed misfit of a dataset applying the focussing coefficients produced by program `focusSpectralKernels`

positional arguments

dmspace_file Data model space input file which defines data and model space.

foc_coef_file Output text file containing the focussing coefficients, as written by program `focusSpectralKernels`

main_parfile Main parameter file of inversion.

3.1.5 computeKernelCoverage

summate the absolute values of the column vectors of a given kernel matrix (for a given number of frequency windows)

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

3.1.6 computeKernels

There are two possible ways to define a set of spectral kernels that are computed:

(way 1): compute kernel for only one path, defined by eventID and station name using options `-evid`, `-staname`, `-comp` and `-param`

(way 2): use flag `-dmspsce` in connection with optional range definition of the path index (flags `-ipath1` `-ipath2`) in order to define a subset of the paths contained in the given data-model-space description. If the upper (lower) limit of the path range is not defined (i.e. `-ipath1` (`-ipath2`) not set), the maximum (minimum) possible value is used. Then, all kernels for the defined range of paths in the given data-model-space description are computed.

positional arguments

main_parfile Main parameter file of inversion.

optional arguments

-evid event_id Defines the event id of the one path (must belong to an event in main event list). (way 1)

-staname station_name Defines the station name of the one path (must belong to a station in main station list). (way 1)

-comp "comp_1 ... comp_n" Vector of receiver components for which the kernel should be computed. For valid components see 1.5 (page 22). (way 1)

-param "param_1 ... param_n" Vector of parameter names for which the kernel should be computed. Only valid parameter names of the chosen model parametrization are accepted. (way 1)

-dmspace dmspace_file Data model space input file to define a set of paths. (way 2)

-ipath1 path1 path1 is the first index of the path loop. By default, `path1 = 1` (way 2)

-ipath2 path2 path2 is the last index of the path loop. By default, `path2 = max_number_of_paths` as to data-model-space description. (way 2)

-wp If set, then plain kernel values on WAVEFIELD POINTS are produced. Otherwise (if not set), pre-integrated kernels on inversion grid cells are computed

3.1.7 computeMisfit

reads in measured and synthetic data characterized by data model space info file and computes the data misfit

positional arguments

dmspace_file Data model space input file which defines data and model space.

main_parfile Main parameter file of inversion.

optional arguments

-jf "jf1..jfn" Vector of *nf* frequency indices. Additionally to the misfit of the whole dataset defined by *dmspace_file*, the misfit is computed for data subsets restricted to these individual frequencies.

3.1.8 createMeasuredData

Fourier transform of time-domain data to ASKI-conform frequency-domain measured data; frequency discretization of measured data as defined in main parfile

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

(see `template/createMeasuredData_parfile_txt_template` for the required parfile for `-txt` input data)

3.1.9 createStartmodelKim

Create a file of type `kernel_inverted_model (.kim)` containing pre-defined values on the inversion grid. Can be used to create a start model for the full waveform inversion process.

Mandatory options

-igtype invgrid_type *invgrid_type* is `TYPE_INVERSION_GRID` as in ASKI iteration step parameter file.

-igpar invgrid_parfile *invgrid_parfile* is `PARFILE_INVERSION_GRID` as in ASKI iteration step parameter file.

-igpath invgrid_path *invgrid_path* is treated as current iteration step path, used for inversion grid to write/read own files.

-mpmtr model_pmtr `model_pmtr` is the model parametrization of the model which is to be created (must be consistent with content of `model_file`, see `-mfile`).

-mtype model_type `model_type` is a string defining the type of `model_file` and the interpolation type:

'1D_linear':

linear interpolation of values between coordinates given in 1D model file.

'3D_structured':

trilinear interpolation of values between coordinates given in 3D structured model file.

1D model file must have the following format

line 1: `nval ncol`

`nval` = number of model values /interpolation coordinates to come

`ncol` = number of columns to read from file, `ncol=1+N`, where `N` is number of parameters

line 2: `icoord param1 ... paramN`

`icoord` = index of inversion grid / wavefield point coordinate for which the interpolation should be applied (either 1,2 or 3)

`param1 ... paramN` = `N` parameter names associated with the values in the columns below

lines 3 ...`nval+2`: `coord val1 ... valN`

these following `nval` lines contain the interpolation coordinate and `N` model values for the respective parameters, as defined by line 2. The values '`coord`' are assumed to be strictly monotonical(!) and can be either increasing or decreasing. You may choose this monotonicity at your will dependent on the coordinate for which the interpolation should be done (which, dependent on your inversion grid type may be depth or positive z-value...)

3D model file must have the following format:

line 1: `nx ny nz`

`nx`, `ny` and `nz` are the number of model points in X-, Y- and Z-direction

line 2: `minX minY minZ`

`minX`, `minY` and `minZ` are the smallest coordinates in the model

line 3: `maxX maxY maxZ`

`maxX`, `maxY` and `maxZ` are the highest coordinates in the mode

`nval`, the total number of vaules in the model is given by `nval = nx*ny*nz`

line 4: `param1 ... paramN`

`param1 ... paramN`, `N` parameter names associated with the values in the columns below

line 5+: nval model values

The model values have to be sorted like:

```
DO i=minX,maxX+1
    DO j=minY,maxY+1
        DO k=minZ,maxZ+1
            READ model value
        END DO
    END DO
END DO
```

-mfile model_file model_file is the model input file of type defined by model_type.

optional arguments

-o outbase outbase is output base name (default is start_model).

-bin If set, the output vtk files will be binary, otherwise they will be ascii.

3.1.10 exportKim

Program exportKim produces a text file (option `-otxt`) containing information of cell centers and radii (i.e. rough expansion) of all inversion grid cells and all cell neighbours, as well as model values and respective invgrid cell indices for each model parameter, as contained in a given kernelInvertedModel file. This text file may be used by any forward method to define the simulation model for the next iteration step, or by any tool handling final models.

The format of the produced text file is as follows:

The first 3 lines contain:

```
model_parametrization
number_of_parameters name_param_1 ... name_param_n
number_of_invgrid_cells
```

The next number_of_invgrid_cells lines contain for each inversion grid cell:

```
c1 c2 c3 r nnb inb1 ... inbn
```

where `c1`, `c2`, `c3` are the 3 coordinates of the cell center (in wavefield point coords), `r` is the cell radius (i.e. rough expansion of cell), `nnb` is the number of cell neighbours and `inb1`, ..., `inbn` are their `nnb` cell indices (if `nnb`>0, otherwise line ends on `nnb`=0). Then number_of_parameters blocks are following in the file, each having the following format:

```

name_param
nval
cell_indx
model_values

```

where `name_param` is the name of the parameter to which the following model values belong, `nval` is the number of model values following, `cell_indx` is a vector of `nval` cell indices to which the model values belong (space separated on one line) and `model_values` is an vector of the actual `nval` model values (space separated on one line).

Additionally (or alternatively) the program converts the `.kim` file to `vtk` files (option `-ovtk`). The two options `-otxt` , `-ovtk` can be used independently of each other.

positional arguments

main_parfile Main parameter file of inversion.

Mandatory options

-kim kernel_inverted_mode_file `kernel_inverted_mode_file` is the binary file containing the inverted model, which is to be exported.

-otxt outfile_txt `outfile_txt` is the text output file to which inversion grid and model information will be written.

-ovtk outfile_vtk `outfile_vtk` is the output file base to which standard `vtk` files of the `.kim` file will be written.

3.1.11 focusSpectralKernels

compute Backus-Gilbert focussing of sensitivity kernels on a defined focussing region in the model space

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

3.1.12 initBasics

Initiating and testing all basic requirements for ASKI programs (parameter files, event and station list, inversion grid, wavefield points, integration weights, reference model)

positional arguments

main_parfile Main parameter file of inversion.

optional arguments

-recr If set, existing files will be recreated with current parfile specifications (overwrites existing files will be read in and not newly created). Affected files are:

- inversion_grid and integration_weights and all related .vtk files (including wavefield_points.vtk files)
- stations.vtk, events.vtk
- kernel reference model files (on wavefield points, and as interpolation on inversion grid)

IF NOT SET (default), EXISTING FILES (especially inversion_grid, integration_weights) WILL BE READ IN ONLY, REGARDLESS OF ANY CHANGES IN PARFILES! So, if you change the specification of inversion grid, or integration weights (or stations, events which is only relevant for station.vtk events.vtk), after having already created the respective files, you should set **-recr**.

3.1.13 investigateDataResiduals

Gets some statistics about the data residuals of a given data (sub)set.

positional arguments

dmsi_file Data-model-space-info file

outdir_stats_files Output directory where residual files per path and component will be written.

main_parfile Main parameter file of inversion.

optional arguments

-ovtk outfile_vtk outfile_vtk of output vtk files. If not set, no vtk files will be produced. (Default = dataset)

3.1.14 invgrid2vtk

Create vtk file(s) of the given inversion grid (useful to look at, to see if the specifications are correct)

Mandatory options

-igtype invgrid_type invgrid_type is TYPE_INVERSION_GRID as in ASKI iteration step parameter file.

-igpar invgrid_parfile invgrid_parfile is PARFILE_INVERSION_GRID as in ASKI iteration step parameter file.

-igpath invgrid_path invgrid_path is treated as current iteration step path, used for inversion grid to write/read own files.

Optional options

-o outbase outbase is output base name (default is inversion_grid)

-overwr If set, existing output files will be overwritten.

-nb "idx_1 ... idx_n Vecotr of n cell indices, indicating a set of cells the neighbours of which will be written as vtk files. IN THE FUTURE: may be helpful to also accept ranges like "20:40".

-all_nb Indicating to use all cell indices to write neighbours for. Must not be set simultaneously along with **-nb**.

-bin If set, the output vtk files will be binary, otherwise they will be ascii.

-recre If set, the existing inversion grid file(s) (if any existing, and if type creates any) will be recreated with current invgrid parfile specifications. If not set (default), existing inversion grid will be read in only, REGARDLESS OF ANY CHANGES IN THE PARFILE!

3.1.15 kdispl2vtk

Extract kernel displacement spectra to vtk files for certain wavefield and strain components and frequencies

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

3.1.16 kernel2vtk

Program kernel2vtk writes the pre-integrated spectral sensitivity kernels as vtk files for specific paths, parameters, components and frequencies.

positional arguments

main_parfile Main parameter file of inversion.

optional arguments

-evid event_id Defines the event id of the one path. (must belong to an event in main event list)

-staname station_name Defines the station name of the one path. (must belong to a station in main station list)

-comp "comp_1 ... comp_n" Vector of receiver components for which vtk files should be generated. For valid components see 1.5 (page 22).

-param "param_1 ... param_n" Vector of parameter names for which vtk files should be generated. Only valid parameter names of the chosen model parametrization are accepted.

-ifreq "idx_1 ... idx_n" Vector of n frequency indices for which vtk files should be generated.

-all_ifreq If set, for all frequency indices of current iteration vtk files will be generated. Must not be set simultaneously along with **-ifreq**.

-wp If set, then the original kernels on the wavefield points are produced (recalculated!) INSTEAD of the pre-integrated ones on the inversion grid.

3.1.17 kgt2vtk

Extract kernel green tensor spectra to vtk files for certain wavefield and strain components and frequencies

This manual section is still to be written, sorry. I hope to provide it soon in a new manual version. Calling the executable without arguments will print a very short help message on how to use it.

3.1.18 krm2kim

interpolate kernel reference model onto inversion grid and produce a .kim file of it

positional arguments

outfile_base Basename of output model files - will additionally be written as vtk.

main_parfile Main parameter file of inversion.

optional arguments

-krm kernel_reference_mode_file If set, then instead of using the kernel reference model file defined by the iteration step parfile, the given file `kernel_reference_mode_file` is used to read in the kernel reference model.

3.1.19 paths2vtk

Plots all paths contained in the data space definition as vtk lines.

positional arguments

dmsi_file Data-model-space-info file

outfile Output file (basename) of the vtk file(s).

main_parfile Main parameter file of inversion.

3.1.20 solveCglsKernelSystem

Solves linear system of sensitivity kernel equations in parallel by a conjugate-gradient method. The algorithm applied here is the "CGLS1" from paper [BES98].

By default, the conjugate-gradient algorithm uses single precision. In case the program crashes due to what looks like precision problems (dividing by 0, getting values "NaN" etc.), you may try to switch to double precision (change the respective line in the very beginning of code file `f90/solveCglsKernelSystem.f90` and recompile `solveCglsKernelSystem`).

This executable is parallelized using MPI. Hence, it should be called like

`mpirun -np 8 solveCglsKernelSystem` (with respective arguments)

The executable uses the maximum number of parallel slots being available in the MPI environment, in the above example this would be 8.

positional arguments

dmsi_file Data-model-space-info file.

outfile_base Base name of output files (will be used for all files, with suitable extensions)

CG_parfile Parameter file defining details related to the conjugate gradient algorithm used to solve the linear system (a template is provided: `template/solveCglsKernelSystem_parfile_template`).

main_parfile Main parameter file of inversion.

optional arguments

-regscal type_regul_scaling `type_regul_scaling` is the type of scaling of regularization constraints, at the moment only `absmax_per_param`, `overall_factor`, `absmax_per_param`, `param_factors` and `none` are supported

-smoothing scaling_values If set, smoothing conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param`, `overall_factor`: one single factor
`absmax_per_param`, `param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-smoothbnd type_smoothing_boundary `type_smoothing_boundary` defines the way (non-existing) neighbours are treated in smoothing conditions at outer/inner boundaries of the inversion grid. Supported types:
`zero_all_outer_bnd`: apply zero smoothing conditions at *all* outer boundaries.
`zero_burried_outer_bnd`, `cont_free_surface`: apply zero smoothing conditions at all outer boundaries *except* on free surfaces.
If not set, standard average is used everywhere (equivalent to continuity boundary conditions).

-damping scaling_values If set, damping conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param`, `overall_factor`: one single factor
`absmax_per_param`, `param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-odir If set, `outfile_base` will be assumed relatively to iteration step output files directory.

-startsol file_name_starting_solution The `.kim` file `file_name_starting_solution` defines the starting solution of linear system.

-normalize type_data_normalization Optionally, `type_data_normalization` defines the type of data normalization. Supported types:
`maxamp_mdata_by_paths`: (description follows)
`maxamp_mdata_by_paths_and_frequency`: (description follows)
`scale_maxamp_mdata_by_paths`: (description follows)

3.1.21 solveKernelSystem

Do inversion step by solving the kernel linear system defined by data model space info and regularization constraints. Serial program using LAPACK library.

positional arguments

dmspace_file Data model space input file which defines data and model space.

outfile Basename of output model files – output model will additionally be written as vtk.

main_parfile Main parameter file of inversion.

optional arguments

-regscal type_regul_scaling `type_regul_scaling` is the type of scaling of regularization constraints, at the moment only `absmax_per_param`, `overall_factor`, `absmax_per_param`, `param_factors` and `none` are supported

-smoothing scaling_values If set, smoothing conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param`, `overall_factor`: one single factor
`absmax_per_param`, `param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-smoothbnd type_smoothing_boundary `type_smoothing_boundary` defines the way (non-existing) neighbours are treated in smoothing conditions at outer/inner boundaries of the inversion grid. Supported types:
`zero_all_outer_bnd`: apply zero smoothing conditions at *all* outer boundaries.
`zero_burried_outer_bnd`, `cont_free_surface`: apply zero smoothing conditions at all outer boundaries *except* on free surfaces.
If not set, standard average is used everywhere (equivalent to continuity boundary conditions).

-damping scaling_values If set, damping conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param`, `overall_factor`: one single factor
`absmax_per_param`, `param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-odir If set, `outfile_base` will be assumed relatively to iteration step output files directory.

-normalize type_data_normalization Optionally, `type_data_normalization` defines the type of data normalization. Supported types:
`maxamp_mdata_by_paths`: (description follows)
`maxamp_mdata_by_paths_and_frequency`: (description follows)
`scale_maxamp_mdata_by_paths`: (description follows)

3.1.22 solveParKernelSystem

Solves linear system of sensitivity kernel equations in parallel by ScaLAPACK libraries.

This executable is parallelized using BLACS libraries which usually are based on MPI. Hence, it should probably be called like

```
mpirun -np 8 solveParKernelSystem (with respective arguments)
```

The executable uses the maximum number of parallel slots being available in the MPI environment, in the above example this would be 8. `solveParKernelSystem` minimally requires `NPROC_ROWS*NPROC_COLUMNS` parallel slots, as defined by the `mpi_parfile`. *If there are less slots available than required, the program will raise an error.*

positional arguments

dmsi_file Data-model-space-info file

outfile_base Base name of output files (will be used for all files, with suitable extensions)

mpi_parfile Parameter file defining everything related to the ScaLAPACK parallelization of the linear system (a template is provided: `template/solveParKernelSystem_parfile_template`).

main_parfile Main parameter file of inversion.

optional arguments

-regscal type_regul_scaling `type_regul_scaling` is the type of scaling of regularization constraints, at the moment only `absmax_per_param`, `overall_factor`, `absmax_per_param`, `param_factors` and `none` are supported

-smoothing scaling_values If set, smoothing conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param, overall_factor`: one single factor
`absmax_per_param, param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-smoothbnd type_smoothing_boundary `type_smoothing_boundary` defines the way (non-existing) neighbours are treated in smoothing conditions at outer/inner boundaries of the inversion grid. Supported types:

`zero_all_outer_bnd`: apply zero smoothing conditions at *all* outer boundaries.

`zero_burried_outer_bnd, cont_free_surface`: apply zero smoothing conditions at all outer boundaries *except* on free surfaces.

If not set, standard average is used everywhere (equivalent to continuity boundary conditions).

-damping scaling_values If set, damping conditions are applied. `scaling_values` is a vector of scaling values consistent with `-regscal`:
`absmax_per_param, overall_factor`: one single factor
`absmax_per_param, param_factors`: one factor per parameter name of current parametrization (in conventional order)
`none` : values given here are ignored

-odir If set, `outfile_base` will be assumed relatively to iteration step output files directory.

-normalize type_data_normalization Optionally, `type_data_normalization` defines the type of data normalization. Supported types:
`maxamp_mdata_by_paths`: (description follows)
`maxamp_mdata_by_paths_and_frequency`: (description follows)
`scale_maxamp_mdata_by_paths`: (description follows)

3.1.23 spec2timeKernels

Compute `time_kernel` files from existing `spectral_kernel` files (by inverse Fourier transform) for the time windows as specified by options `t0, dt, nt1, nt2`: The total set of time samples consists of `n` time windows. `nt1, nt2` are strings containing `n` integers defining the start (`nt1`) and end (`nt2`) each time window by a time sample index. Times compute as $t = t_0 + j t \cdot dt$ where $nt1 \leq j t \leq nt2$. There are two possible ways to define a set of kernels that are transformed:

- (way 1): compute kernel for only one path, defined by `eventID` and station name using options `-evid, -staname, -comp` and `-param`
- (way 2): use flag `-dmsapce` in connection with optional range definition of the path index (flags `-ipath1 -ipath2`)
in order to define a subset of the paths contained in the given data-model-space description. If the upper (lower) limit of the path range is not defined (i.e. `-ipath1 (-ipath2)` not set), the maximum (minimum) possible value is used. Then, all kernels for the defined range of paths in the given data-model-space description are computed.

positional arguments

main_parfile Main parameter file of inversion.

optional arguments

set of kernels to be transformed

way 1

-evid event_id Defines the event id of the one path. (must belong to an event in main event list)

-staname station_name Defines the station name of the one path. (must belong to a station in main station list)

-comp "comp_1 ... comp_n" Vector of receiver components for which time kernel should be transformed. For valid components see 1.5 (page 22).

-param "param_1 ... param_n" Vector of parameter names for which time kernel should be transformed. Only valid parameter names of the chosen model parametrization are accepted.

way 2

-dmspace Data model space input file to define a set of paths.

-ipath1 path1 path1 is the first index of the path loop. By default, `path1 = 1`

-ipath2 path2 path2 is the last index of the path loop. By default, `path2 = max_number_of_paths` as to data-model-space description.

-wp If set, then 'ON-WP' spectral kernel files are produced, containing plain kernel values on wavefield points. If not set, normal kernel files (pre-integrated) are transformed.

time waveform kernels

-dt time_step Global time step of time discretization.

-nt1 "idx_1 .. idx_n" Vector of `n` time indices defining the start indices of the `n` time windows.

-nt2 "idx_1 .. idx_n" Vector of `n` time indices defining the start indices of the `n` time windows.

-t0 tzero Optional global time shift which is added to all times defined by `dt`, `nt1`, `nt2` (default `t0=0`)

3.1.24 timeKernel2vtk

Produces `vtk` files from *one existing* binary time sensitivity kernel file for a selection of time steps defined by vectors of starting and end indices `-nt1`, `-nt2`

positional arguments

main_parfile Main parameter file of inversion.

optional arguments

-evid event_id Defines the event id of the one path. (must belong to an event in main event list)

-staname station_name Defines the station name of the one path. (must belong to a station in main station list)

-comp "comp_1 ... comp_n" Vector of receiver components for which vtk files should be generated. For valid components see 1.5 (page 22).

-param "param_1 ... param_n" Vector of parameter names for which vtk files should be generated. Only valid parameter names of the chosen model parametrization are accepted.

-nt1 "idx_1 .. idx_n" Vector of n time indices defining the start indices of the n time windows within which vtk files should be generated (the resulting set of time indices must be contained in the binary time kernel files generated by `spec2timeKernels`).

-nt2 "idx_1 .. idx_n" Vector of n time indices defining the start indices of the n time windows within which vtk files should be generated (the resulting set of time indices must be contained in the binary time kernel files generated by `spec2timeKernels`).

3.2 Integration Weights

The ASKI module `integrationWeights` computes integration weights for the set of wavefield points in order to integrate the kernels over the inversion grid. As we need to calculate the integrals of the kernels over each inversion grid cell separately, the integration weights are computed for each cell in such a way that weighting the summation of the kernel values yields the desired integral value:

For each inversion grid cell $\Omega_c \subset \mathbb{R}^3$ which contains wavefield points $\mathbf{x}_1, \dots, \mathbf{x}_{n_c}$ the weights w_1, \dots, w_{n_c} are computed such that

$$\int_{\Omega_c} K(\mathbf{x}) d\mathbf{x} \simeq \sum_{i=1}^{n_c} w_i K(\mathbf{x}_i) \quad (3.1)$$

There are several types of integration weights supported (indicated by dummy variable `intw_type` of subroutine `createIntegrationWeights`):

3.2.0 Compute Average (no integration)

In case of `intw_type = 0`, function `createIntegrationWeights` sets

$$w_i = \frac{1}{n_c}, \quad i = 1, \dots, n_c$$

in each inversion grid cell Ω_c .

This way, the summation $\sum_{i=1}^{n_c} w_i K(\mathbf{x}_i^G) = \frac{1}{n_c} \sum_{i=1}^{n_c} K(\mathbf{x}_i^G)$ yields the average kernel value in Ω_c .

This type of integration weights (which are actually no integration weights) may be used to perform some sort of interpolation of kernel values onto the inversion grid (e.g. in order to compare kernel values from different methods which use different sets of wavefield points).

3.2.1 Scattered Data Integration

In case of `intw_type = 1...4`, a method by David Levin [Lev99] is applied to a standardized inversion grid cell Ω^S . For different shapes of inversion grid cells, different types of standard cells are used, which are referred to below.

For each inversion grid cell $\Omega_c \subset \mathbb{R}^3$ containing wavefield points $\mathbf{x}_1, \dots, \mathbf{x}_{n_c}$, a transformation $T : \Omega_c \rightarrow \Omega^S$ is used to transform cell Ω_c into the standard cell Ω^S and to compute the respective transformed wavefield points $\mathbf{x}_i^S = T(\mathbf{x}_i)$ contained in Ω^S .

Then [Lev99] is applied to points $\mathbf{x}_1^S, \dots, \mathbf{x}_{n_c}^S$ and volume Ω^S to compute integration weights $w_1^S, \dots, w_{n_c}^S$ such that

$$\begin{aligned} \int_{\Omega_c} K(\mathbf{x}) d\mathbf{x} &= \int_{\Omega^S} K(T^{-1}(\mathbf{x}^S)) \mathcal{J}_{T^{-1}}(\mathbf{x}^S) d\mathbf{x}^S \\ &\simeq \sum_{i=1}^{n_c} w_i^S \mathcal{J}_i K(\mathbf{x}_i) \\ &= \sum_{i=1}^{n_c} w_i K(\mathbf{x}_i) \end{aligned} \quad (3.2)$$

where $\mathcal{J}_{T^{-1}}$ denotes the Jacobian of the inverse transformation T^{-1} , $\mathcal{J}_i = \mathcal{J}_{T^{-1}}(\mathbf{x}_i^S)$ and the desired weights compute as $w_i = w_i^S \mathcal{J}_i$. The method of computing such integration weights $w_1^S, \dots, w_{n_c}^S$, as presented in [Lev99], is explained in the following.

The Method of Scattered Data Integration

[Lev99] follows a composite rule strategy for building the integration weights. For subsets of the volume of interest it constructs integration formulae which are as local and as stable as possible and are exact for polynomials p of a certain fixed degree m . It is assumed that the integrals of these polynomials $p \in \Pi_m$ over the subsets are easily computable.

In notation of [Lev99], the integration weights A_i for a function f on domain $\Omega \subset \mathbb{R}^d$ which is given on a set $\{x_i\}_{i=1}^N \subset \Omega$ are constructed as

$$A_i = \sum_{k=1}^K A_i^{(k)}, \quad 1 \leq i \leq N,$$

where Ω is subdivided into K disjoint subsets E_k . For each E_k , the N weights $A_i^{(k)}$ are calculated as follows.

We choose a basis $\{p_i\}_{i=1}^J$ of the space Π_m of all polynomials in \mathbb{R}^d with maximum total degree m , where $J = \binom{d+m}{m}$ is the dimension of space Π_m . $A_i^{(k)}$ are then defined as the components

$a_i = A_i^{(k)}$ of vector $\bar{a} = D^{-1} E (E^t D^{-1} E)^{-1} \bar{c}$, where

$$D = 2\text{Diag} \{ \eta(\|x^* - x_1\|), \dots, \eta(\|x^* - x_N\|) \}$$

$$E_{i,j} = p_j(x_i), \quad 1 \leq i \leq N, \quad 1 \leq j \leq J$$

and \bar{c} contains the integrals of the p_i over E_k , i.e. $c_i = \int_{E_k} p_i$. $\eta(r) = \exp(r^2/h^2)$ is a fast increasing weight function which gives the localizing properties of the weights. h is approximately the diameter of subsets E_k and x^* is some center of E_k .

This composite local approach of calculating global integration weights involves K solutions of a full linear system of order J .

Application to Hexahedral Inversion Grid Cells

For inversion grid cells of general hexahedral shape, the 3-dimensional cube

$$\Omega^S = [-1, 1]^3 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid -1 \leq x, y, z \leq 1 \right\}$$

is used as the standard cell. For every such inversion grid cell Ω_c , module `inversionGrid` is expected to provide its transformed wavefield points $\mathbf{x}_1^S, \dots, \mathbf{x}_{n_c}^S$ and their corresponding values of Jacobian \mathcal{J}_i .

In the context of Scattered Data Integration, the inversion domain $\Omega = \Omega^S = [-1, 1]^3$ is subdivided into $K = n_h^3$ subcubes E_k of edge length $h = 2/n_h$. $n_h = \max \left\{ \left\lfloor \sqrt[3]{\frac{n_c}{J}} \right\rfloor, 1 \right\}$ is chosen in such a way that there should be at least J (or all, otherwise) integration points within E_k , as otherwise the damping by matrix D^{-1} might cause numerical instabilities by making matrix $E^t D^{-1} E$ close to singular.

As x^* , the center of the respective subcube is chosen.

The desired weights $w_1^S, \dots, w_{n_c}^S$ are then given by $w_i^S = A_i$, $1 \leq i \leq n_c$.

Application to Tetrahedral Inversion Grid Cells

For inversion grid cells of general tetrahedral shape, the 3-dimensional simplex with corners

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

is used as the standard cell Ω^S . For every such inversion grid cell Ω_c , module `inversionGrid` is expected to provide its transformed wavefield points $\mathbf{x}_1^S, \dots, \mathbf{x}_{n_c}^S$ and their corresponding values of Jacobian \mathcal{J}_i .

In the context of Scattered Data Integration, here the inversion domain $\Omega = \Omega^S$ is *not* subdivided into any true subsets E_k . It is always $K = 1$ and $E_1 = \Omega$, mainly because a subdivision of the standard tetrahedron is not trivial (compared with e.g. the cube $[-1, 1]^3$), considering that the integrals of the base polynomials must be computed over all subsets E_k .

As x^* , the barycenter

$$\begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}$$

of the standard simplex is chosen and $h = 1$.

The desired weights $w_1^S, \dots, w_{n_c}^S$ are then given by $w_i^S = A_i$, $1 \leq i \leq n_c$

Scattered Data Integration, Order 1

`intw_type = 1`

In the context of this subsection 3.2.1, $m = 1$ is used as the degree of polynomials which are integrated in an exact way and of course $d = 3$. The space Π_1 of all polynomials in \mathbb{R}^3 of maximum total degree $m = 1$ has dimension $J = \binom{3+m}{m} = \binom{4}{1} = 4$. As a basis of Π_1 we choose $\{1, x, y, z\}$.

Scattered Data Integration, Order 2

`intw_type = 2`

In the context of this subsection 3.2.1, $m = 2$ is used as the degree of polynomials which are integrated in an exact way and of course $d = 3$. The space Π_2 of all polynomials in \mathbb{R}^3 of maximum total degree $m = 2$ has dimension $J = \binom{3+m}{m} = \binom{5}{2} = 10$. As a basis of Π_2 we choose $\{1, x, y, z, x^2, xy, xz, y^2, yz, z^2\}$.

Scattered Data Integration, Order 3

`intw_type = 3`

In the context of this subsection 3.2.1, $m = 3$ is used as the degree of polynomials which are integrated in an exact way and of course $d = 3$. The space Π_3 of all polynomials in \mathbb{R}^3 of maximum total degree $m = 3$ has dimension $J = \binom{3+m}{m} = \binom{6}{3} = 20$. As a basis of Π_3 we choose $\{1, x, y, z, x^2, xy, xz, y^2, yz, z^2, x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3\}$.

Scattered Data Integration, Optimal Order

In case of `intw_type = 4`, function `createIntegrationWeights` tries to separately find for each inversion grid cell the highest possible order of Scattered Data Integration. Starting with highest order $m = 3$, it continues to recompute Scattered Data Integration weights of order $m = 2$ and $m = 1$ until the computation was successful. If the computation for order $m = 1$ fails, the integration weights of that cell will be marked erroneous, the computation of the weights is not successful in that case.

As the success of the Scattered Data Integration method is strongly dependent on the specific set of points $\mathbf{x}_1^S, \dots, \mathbf{x}_{n_c}^S$, since matrix $E_{i,j} = p_j(x_i)$ must have full rank, the strategy of choosing the highest possible degree of integration for each cell tries to take all locally available information of inversion grid and wavefield points into account.

3.2.2 Linear (first order) Integration

In case of `intw_type = 5`, function `createIntegrationWeights` sets

$$w_i = \frac{1}{n_c} \text{vol}(\Omega_c) \quad , \quad i = 1, \dots, n_c$$

in each inversion grid cell Ω_c , where $\text{vol}(\Omega_c)$ denotes the volume of inversion grid cell Ω_c , which is expected to be provided by module `inversionGrid` for every cell.

This way, the summation $\sum_{i=1}^{n_c} w_i K(\mathbf{x}_i^G) = \text{vol}(\Omega_c) \frac{1}{n_c} \sum_{i=1}^{n_c} K(\mathbf{x}_i^G)$ yields the average kernel value in Ω_c multiplied with the volume of Ω_c .

This somehow approximates the generalization of the trapezoidal rule to 3 dimensions, in which the integral of a function f over some tetrahedron \mathcal{T} , which is defined by 4 incoplanar points $\mathbf{t}_1, \dots, \mathbf{t}_4$, is computed by $\text{vol}(\mathcal{T}) \frac{1}{4} \sum_{i=1}^4 f(\mathbf{t}_i)$.

3.2.3 External Integration Weights

In case of `intw_type = 6`, function `createIntegrationWeights` does not actually compute any integration weights. Instead, it calls function `transformToStandardCellInversionGrid` of module `inversionGrid` with dummy variable `type_standard_cell` set to value `-1`, which requests the routine to return the total integration weights in variable `jacobian` instead the jacobian values. These returned values are then stored as the integration weights.

This functionality must be supported by the type of inversion grid. At the moment only inversion grids of type `specfem3dInversionGrid` support external type integration weights.

Bibliography

- [BES98] Åke Björck, Tommy Elfving, and Zdeněk Strakoš. Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM Journal on Matrix Analysis and Applications*, 19(3):720–736, 1998.
- [FD95] W. Friederich and Jörg Dalkolmo. Complete synthetic seismograms for a spherically symmetric earth by a numerical computation of Green’s function in the frequency domain. *Geophys. J. Int.*, 122:537–550, 1995.
- [FS06] Florian Schumacher. *Modularized iterative full seismic waveform inversion for 3D-heterogeneous media based on waveform sensitivity kernels*. PhD thesis, 2014-02-06. urn:nbn:de:hbz:294-40511.
- [Lev99] D. Levin. Stable integration rules with scattered integration points. *Journal of Computational and Applied Mathematics*, 112:181–187, 1999.
- [Sch15] Florian Schumacher. *Using SPECFEM3D_Cartesian-3.0 for ASKI– Analysis of Sensitivity and Kernel Inversion, version 1.0*, 2015.
- [SFL 8] Florian Schumacher, Wolfgang Friederich, and Samir Lamara. A flexible, extendable, modular and computationally efficient approach to scattering-integral-based seismic full waveform inversion. *Geophysical Journal International*, Accepted 2015 November 17. Received 2015 November 17; in original form 2015 April 8. <http://dx.doi.org/10.1093/gji/ggv505>.
- [TKL08] Jeroen Tromp, Dimitri Komatitsch, and Qinya Liu. Spectral-element and adjoint methods in seismology. *Communications in Computational Physics*, 3(1):1–32, 2008.

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

`<http://fsf.org/>`

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within

that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the

History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of

Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.