# **SeisFlows Documentation**

Release 0.1

# **CONTENTS**

1	Overview	1
2	Usage 2.1 Installation	3 3 4 5
3	Executable 3.1 Scripts	<b>7</b> 7
4	Designing Project 4.1 Parameter Files: parameters.py	9 10
5	<u> </u>	13 13 13
6	Module6.1seisflows.optimize package6.2seisflows.plugins package6.3seisflows.postprocess package6.4seisflows.preprocess package6.5seisflows.solver package6.6seisflows.system package6.7seisflows.tools package6.8seisflows.workflow package6.9seisflows.config module	15 17 23 24 26 30 37 41 45
Рy	thon Module Index	47
In	dex	49

#### ONE

#### **OVERVIEW**

SeisFlows is a Python waveform inversion package with a growing user base in academia and industry. So far, the package has been used for production runs with a billion or so model parameters and for research on oil and gas exploration, earthquake seismology, and general nonlinear optimization problems.

To provide flexibility, SeisFlows is very modular. Users are offered choices in each of the following categories:

- · workflow
- system
- · solver
- · pre-processing
- · post-processing
- nonlinear optimization

The thing that ties everything together is the workflow. Execution of a workflow is equivalent to stepping through the code contained in workflow.main. Users are free to customize the default 'inversion' and 'migration' workflows from the main package.

A number of options exist for system and solver. By isolating system and solver machinery, users can switch from one application to another with relative ease. For example, if the study area in an earthquake tomography project expands, users can trade a regional Cartesian solver for a global solver. If a PBS cluster goes offline and a SLURM cluster comes online to replace it, users can trade the PBS system interface for a SLURM system interface.

A selection of ready-to-go system and solver interfaces is provided in the main package. Through these interfaces, SeisFlows (or prototypes of it) have run on clusters managed by the Department of Defense, Chevron Corp., Total S.A., Princeton University and other universities and institutions.

Users can also choose from various pre-processing and post-processing options. In our terminology, pre-processing consists of signal processing operations performed on seismic traces prior to the gradient computation. Post-processing consists of regularization or image processing operations carried out after the gradient computation.

If desired functionality is missing from the main package, users can contribute their own classes or overload default ones.

**CHAPTER** 

**TWO** 

**USAGE** 

#### 2.1 Installation

To install Seisflows, first clone the repository:

```
git clone https://github.com/rmodrak/seisflows.git
```

Then set environment variables. Add the following lines to .bashrc (or modify accordingly if using a shell other than bash):

```
export PATH=$PATH:/path/to/seisflows/scripts
export PYTHONPATH=$PYTHONPATH:/path/to/seisflows
```

## 2.1.1 Software Prerequisites

SeisFlows requires Python 2.7, Numpy, Scipy and Obspy. Users will need to install these packages before being able to use SeisFlows.

Forward modeling software is also a prerequisite; see Solver Configuration for more information.

#### 2.1.2 Hardware Prerequisites

Access to a computer cluster is required for most applications. Base classes are provided for several common cluster configurations, including PBS and SLURM. Nonstandard configurations can often be accommodated through modifications to one of the base classes; see *System Configuration* for details.

#### 2.2 Job Submission

Each job must have it own working directory within which users must supply two input files, paths.py and parameters.py.

To begin executing a workflow, simply type sfrun within a working directory. If an inversion workflow and serial system configuration, for example, are specified in the parameters file, the inversion will begin executing immediately in serial. If a PBS, SLURM, or LSF system configuration is specified instead, execution may wait until required resources become available.

Once the workflow starts running, status information is displayed to the terminal or to the file output.log. By default, updated models and other inversion results are output to the working directory.

# 2.3 Solver Configuration

SeisFlows includes Python interfaces for SPECFEM2D, SPECFEM3D, and SPECFEM3D\_GLOBE. While the Python interfaces are part of the SeisFlows package, the solver source code must be downloaded separately through GitHub.

After downloading the solver source code, users must configure and compile it, following the instructions in the solver user manual. Summarized briefly, the configuration and compilation procedure is:

Prior to compilation, users need to run the configure script and prepare input files such as

- · parameter file
- · source file
- · stations file.

To successfully run the configure, you may need to install compilers, libraries, and other software in your environment.

The result of compilation is a set of binary executables, such as

- mesher
- · solver
- · smoothing utility
- · summing utility.

After compilation, solver input files must be gathered together in one directory and solver executables in another. The absolute paths to the directories containing input files and executables must be given in paths.py.

## 2.3.1 Writing Custom Solver Interfaces

Besides SPECFEM2D, SPECFEM3D, and SPECFEM3D\_GLOBE, SeisFlows can interface with other solvers capable of running forward and adjoint simulations. Users unaffiliated with the main SeisFlows developers have succeeded in interfacing with, for example, their own finite difference solvers. For information about writing custom solver interfaces, see *Developer Reference*.

## 2.3.2 Design Philosophy

Integration of the solver with the other workflow components can be challenging. Here we try to give an idea of the issues involved from both a developer and a user standpoint.

- Solver computations account for most of the cost of an inversion. As a result, the solver must be written in an
  efficient compiled language, and wrappers must be written to integrate the compiled code with other software
  components.
- There is currently no mechanism for automatically compiling executables for SPECFEM2D, SPECFEM3D, or SPECFEM3D\_GLOBE. Users must prepare their own SPECFEM input files and then follow the compilation procedure in the SPECFEM documentation.
- As described above *Job Submission*, SeisFlows uses two input files, paths.py and parameters.py. Problems could arise if parameters from SeisFlows input files conflict with parameters from solver input file. Users must make sure that there are no conflicts between SeisFlows parameters and solver parameters.
- In the solver routines, it's natural to represent velocity models as dictionaries, with different keys corresponding to different material parameters. In the optimization routines, it's natural to represent velocity models as vectors. To convert back and forth between these two representations, a pair of utility functions—split and merge—are included in solver.base.

4 Chapter 2. Usage

# 2.4 System Configuration

SeisFlows can run on SLURM, PBS, and LSF clusters, as well as, for very small problems, laptops or desktops. A list of available system interface classes follows. By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

*PBS\_SM* - For small inversions on PBS clusters. All resources are allocated at the beginning and all simulations are run within a single job. Requires that individual wavefield simulations run each on a single core, making this option suitable for small 2D inversions only.

*PBS\_LG* - For large inversions on PBS clusters. The work of the inversion is divided between multiple jobs that are coordinated by a single long-running master job. Resources are allocated on a per simulation basis. Suitable for small to medium 3D inversions in which individual wavefield simulation span several or more nodes.

*SLURM\_SM* - For small inversions on SLURM clusters. All resources are allocated at the beginning and all simulations are run within a single job. Requires that each individual wavefield simulation runs only a single core, making this option suitable for small 2D inversions only.

*SLURM\_LG* - For large inversions on SLURM clusters. The work of the inversion is divided between multiple jobs that are coordinated by a single long-running master job. Resources are allocated on a per simulation basis. Suitable for 3D inversions in which individual wavefield simulation span several or more nodes.

SERIAL - Tasks that are normally carried out in parallel are instead carried out one at a time. Useful for debugging, but not much else.

*MULTITHREADED* - On desktops or laptops with multiple cores, allows embarrassingly parallel tasks to be carried out several at a time, rather than one at a time. Can be used to run small 2D inversions on a laptop or desktop.

LSF\_SM - Same as SLURM\_SM and PBS\_SM, except for LSF clusters.

LSF LG - Same as SLURM LG and PBS LG, except for LSF clusters.

#### 2.4.1 Writing Custom System Interfaces

If your needs are more specialized, please view seisflows.system source code to get a sense for how to write your own custom system interfaces. In our experience, system interfaces require no more than a few hundred lines of code, so writing your own is generally possible once you are familiar with the SeisFlows framework and your own cluster environment.

## 2.4.2 Design Philosophy

To make SeisFlows work across different environments, our approach is to wrap system commands with a thin Python layer. To handle job submission, for example, we wrap the PBS command qsub and the SLURM command sbatch with a python utility called *system.submit*. The result is a consistent python interface across different clusters.

Filesystem settings can be adjusted by modifying values in the PATH dictionary, which is populated from paths.py. Output files and temporary files, by default, are written to the working directory. If a value for PATH.SCRATCH is supplied, temporary files are written there instead. If each compute node has its own local filesystem, a value for PATH.LOCAL can be supplied so that temporary files required only for a local process need not be written to the global filesystem.

As the size of an inversion grows, scalability and fault tolerance become increasingly important. If a single forward simulation spans more than one node, users must select pbs\_lg or slurm\_lg system configurations in parameters.py. If a forward simulation fits onto a single node, users should select pbs\_sm or slurm\_sm instead.

In SeisFlows, the overall approach to solving system interface problems is to use lightweight Python wrappers. For complex cluster configurations, heavier-weight solutions may be required. Users are referred to SAGA or Pegasus projects for ideas.

# 2.5 Developer Reference

To allow classes to work with one another, each must conform to an established interface. This means certain classes must implement certain methods, with specified input and output. Required methods include

- setup methods are generic methods, called from the main workflow script and meant to provide users the flexibility to perform any required setup tasks.
- check methods are the default mechanism for parameter declaration and checking and are called just once, prior to a job being submitted through the scheduler.

Besides required methods, classes may include any number of private methods or utility functions.

6 Chapter 2. Usage

## **THREE**

## **EXECUTABLE**

Seisflows provides several executable Python scripts in /path/to/seisflows/scripts. As the PATH has already been added to environmental varibales, thereby can be executed in any directory.

Tests checking module importing, system, and optimization are also provided in /path/to/seisflows/tests. It's recommmended to past these tests prior to other projects.

# 3.1 Scripts

- sfrun (sfsubmit): Workflow submission scripts.
- plotgll: Plots GLL model read from SPECFEM2D Fortran binary file
- sfclean: Delete directories output, output.stat, output.optim and scratch in the current directory
- sfresume: Resume current submitted workflow.
- sfexample: Run seisflows example. Currently not available.

#### 3.2 Tests

- run\_test\_system: Test parallelizaiton environment.
- run\_test\_import: Testing seisflows module importing.
- run\_test\_optimize:
- run\_test\_preprocess:
- run\_test\_tools:

#### **DESIGNING PROJECT**

Before submitting jobs to the local server, parameters defined in parameters.py and path.py require a careful check, avoiding any unnecessary workload.

## 4.1 Parameter Files: parameters.py

parameters.py contains a list of parameter names and values. Prior to a job being submitted, parameters are checked so that errors can be detected without loss of queue time or wall time. Parameters are stored in a dictionary that is accessible from anywhere in the Python code. By convention, all parameter names must be upper case. Parameter values can be floats, integers, strings or any other Python data type. Parameters can be listed in any order.

General

TITLE Project title.

WORKFLOW Workflow specified for seisflows. 'inversion' and 'migration' are currently supported.

**SOLVER** Time domain solver specified for seisflows. See *Solver Configuration* and *seisflows.solver package* for supported options.

**SYSTEM** System type supported for seisflows. See *System Configuration* and *seisflows.system package* for supported options..

**OPTIMIZE** Optimization method used for inversion. See *seisflows.optimize package* for supported options.

**PREPROCESS** Preprocessing workflow specified. See *seisflows.preprocess package* for supported options.

**POSTPROCESS** Postprocessing workflow specified. 'base' needs to be specified.

**MISFIT** Type of misfit for evaluation. See *seisflows.plugins.misfit module* for supported options.

**MATERIALS** Materials of simulation domain. 'Elastic' and 'Acoustic' are currently supported. See seisflows.solver.base.

Workflow

**BEGIN** First iteration index.

**END** Last iteration index.

NREC Number of receivers.

**NSRC** Number of sources. SEM source file needs to be stored with a six-digit index suffix.

**SAVEMODEL** Frequency of saving model. 1 by default.

**SAVEGRADIENT** Frequency of saving gradient. 1 by default.

**SAVEKERNELS** Frequency of saving kernels. 0 by default.

**SAVETRACES** Frequency of saving traces. 0 by default.

**SAVERESIDUALS** Frequency of saving residuals. 0 by default.

Preprocessing

**FORMAT** Data file format.

CHANNELS Data channels. Currenly, 'su', or 'SU' need to be specified.

**NORMALIZE** Apply normalization for traces. 'NormalizeEventsL1', 'NormalizeEventsL2', 'NormalizeTracesL2' are currently supported. See seisflows.preprocess.

Filter

**BANDPASS** Boolean type bandpass switch for traces.

**FILTER** Type of filter used. See *seisflows.preprocess package* for supported options.

**FREQMIN** Low frequency corner.

**FREQMAX** High frequency corner.

Mute

**MUTE** List type switch for trace mute. *seisflows.preprocess package* for supported options.

Postprocessing

**SMOOTH** Smoothing radius. See xsmooth sem for usage.

Optimization

**PRECOND** Preconditioner type. See *Path Files: paths.py* and *seisflows.plugins.preconds.diagonal module*.

STEPMAX Maximum trial steps

Solver

**NT** Number of time steps defined in Par\_file.

**DT** Time step defined in Par\_file.

**F0** Dominant frequency defined in SOURCE.

System

**NTASK** Number of tasks submitted. Currently, **NTASK** must satisfy 1 <= NTASK <= NSRC.

**NPROC** Number of processors.

**MPIEXEC** MPI executable prefix, e.g., mpirun -np 13. Note for a space at the end of the string, as seisflows concatenates the prefix with SPECFEM executable command.

# 4.2 Path Files: paths.py

paths.py contains a list of path names and values. Prior to a job being submitted, paths are checked so that errors can be detected without loss of queue time or wall time. Paths are stored in a dictionary that is accessible from anywhere in the Python code. By convention, all names must be upper case, and all values must be absolute paths. Paths can be listed in any order.

- **DATA** PATH contains seismic data if field data is used for inversion. Data of difference sources should be stored in separate folder. If DATA directory does not exist, seisflows would automatically generate synthetic data using model from MODEL\_TRUE.
- **MODEL\_INIT** PATH contains model file for initial iteraion.
- MODEL\_TRUE PATH contains true model for generating synthetic data.
- **PRECOND** PATH to user supplied diagonal preconditioner. Seisflows will rescale model parameters based on user supplied weights. See *seisflows.plugins.preconds.diagonal module*.
- **MASK** PATH to mask file for gradient scaling. Mask needs to be stored mimicking the file format in which models are stored.
- **SPECFEM\_DATA** PATH to SPECFEM DATA directory which contians Par\_file, SOURCE, and other necessary inputs.
- **SPECFEM\_BIN** PATH to SPECFEM bin directory which contains binary executable command of SPECFEM solver.

**CHAPTER** 

**FIVE** 

## **EXAMPLE**

# 5.1 Examples: Available For Download

We have prepared a 2D waveform inversion example that is inexpensive enough to run on almost any laptop, desktop, or cluster.

Some additional examples are available for download. Please review the instructions for the 2D checkerboard test case to get a sense for how to run these other inversions.

Some 2D examples based on the Marmousi model are available here.

A 3D Cartesian checkerboard example is available here.

A 3D global 1-chunk example is available here. Please note, the compressed archive for this example is very large (> 0.5 GB). [No longer available because of file size.]

At a minimimum, one processer is required for the 2D Marmousi examples, 16 processors are required for the 3D Cartesian example, and 64 processors are required for the global 1-chunk example. See here for more information about running inversions in parallel.

Note: File hosting services are provided by my alma mater. The download server may become temporarily unavailable due to system maintenance or permanently unavailable due to expiration of my account.

# 5.2 Examples: Available Locally

Users with accounts on "tiger.princeton.edu" can run the following inversions without having to download files or recompile executables.

2D Regional and Global

- · North America
- Southern California
- Global
- Deep Earth

#### 2D Near Surface

- Marmousi offshore
- · Marmousi onshore
- · overthrust offshore
- · overthrust onshore

# SeisFlows Documentation, Release 0.1

- BP anticline
- BP salt diapir

#### 3D Cartesian

• checkerboard

#### 3D Global

• mideast

**CHAPTER** 

SIX

#### **MODULE**

# 6.1 seisflows.optimize package

## 6.1.1 seisflows.optimize.base module

```
class seisflows.optimize.base.base
    Bases: object
```

Nonlinear optimization abstract base class

Base class on top of which steepest descent, nonlinear conjugate, quasi- Newton and Newton methods can be implemented. Includes methods for both search direction and line search.

To reduce memory overhead, vectors are read from disk rather than passed from calling routines. For example, at the beginning of compute\_direction the current gradient is read from 'g\_new' and the resulting search direction is written to 'p\_new'. As the inversion progresses, other information is stored as well.

**Variables** m\_new - current model m\_old - previous model m\_try - line search model f\_new - current objective function value f\_old - previous objective function value f\_try - line search function value g\_new - current gradient direction g\_old - previous gradient direction p\_new - current search direction p\_old - previous search direction

```
check()
```

Checks parameters, paths, and dependencies

```
compute_direction()
```

Computes search direction

dot(x, y)

Computes inner product between vectors

#### finalize\_search()

Prepares algorithm machinery and scratch directory for next model upate

#### initialize search()

Determines first step length in line search

load (filename)

loadtxt (filename)

 $model\_cutoff(m)$ 

restart()

Restarts nonlinear optimization algorithm

Keeps current position in model space, but discards history of nonlinear optimization algorithm in an attempt to recover from numerical stagnation

```
retry_status()
```

Determines if restart is worthwhile

After failed line search, determines if restart is worthwhile by checking, in effect, if search direction was the same as gradient direction

```
save (filename, array)
savetxt (filename, scalar)
setup()
    Sets up nonlinear optimization machinery
update_search()
    Updates line search status and step length
    Status codes status > 0: finished status == 0: not finished status < 0: failed</pre>
```

#### 6.1.2 seisflows.optimize.LBFGS module

```
class seisflows.optimize.LBFGS.LBFGS
   Bases: seisflows.optimize.base.base
   Limited memory BFGS algorithm
   check()
        Checks parameters, paths, and dependencies
   compute_direction()
        Computes search direction
   restart()
        Restarts nonlinear optimization algorithm
        Keeps current position in model space, but discards history of nonlinear optimization algorithm in an attempt to recover from numerical stagnation
```

#### 6.1.3 seisflows.optimize.NLCG module

Sets up nonlinear optimization machinery

setup()

```
class seisflows.optimize.NLCG.NLCG
Bases: seisflows.optimize.base.base
Nonlinear conjugate gradient method
check()
        Checks parameters, paths, and dependencies
compute_direction()
        Computes search direction
restart()
        Restarts nonlinear optimization algorithm
        Keeps current position in model space, but discards history of nonlinear optimization algorithm in an attempt to recover from numerical stagnation
setup()
        Sets up nonlinear optimization machinery
```

#### 6.1.4 seisflows.optimize.steepest descent module

```
class seisflows.optimize.steepest_descent.steepest_descent
     Bases: seisflows.optimize.base.base
     Steepest descent method
     check()
          Checks parameters, paths, and dependencies
     compute_direction()
          Computes search direction
     restart()
         Restarts nonlinear optimization algorithm
          Keeps current position in model space, but discards history of nonlinear optimization algorithm in an
          attempt to recover from numerical stagnation
     restarted = False
     setup()
          Sets up nonlinear optimization machinery
6.2 seisflows.plugins package
```

### 6.2.1 seisflows.plugins.line search package

#### seisflows.plugins.line search.backtrack module

```
class seisflows.plugins.line_search.backtrack.Backtrack(step_count_max=10,
                                                                           step_len_max=inf,
                                                                           path='/Users/niyiyu/Documents/GitHub/seisflows/c
     Bases: seisflows.plugins.line_search.bracket.Bracket
     Implements backtracking linesearch
     Variables x - list of step lenths from current line search f - correpsonding list of function values gtg - dot
          product of gradient with itself gtp - dot product of gradient and search direction
     Status codes status > 0: finished status == 0: not finished status < 0: failed
     calculate_step()
          Determines step length and search status
```

#### seisflows.plugins.line\_search.base module

```
class seisflows.plugins.line_search.base.Base(step_count_max=10, step_len_max=inf,
                                                               path='/Users/niyiyu/Documents/GitHub/seisflows/docs')
     Bases: object
     Abstract base class for line search
     Variables x - list of step lenths from current line search f - correpsonding list of function values m - how many
           step lengths in current line search? n - how many model updates in optimization problem? gtg - dot
           product of gradient with itself gtp - dot product of gradient and search direction
```

**Status codes** status > 0: finished status == 0: not finished status < 0: failed

```
calculate_step()
     clear_history()
          Clears line search history
     initialize (step_len, func_val, gtg, gtp)
     search history(sort=True)
          A convenience function, collects information needed to determine search status and calculate step length
     update (step_len, func_val)
class seisflows.plugins.line_search.base.Writer(path='./output.optim')
     Bases: object
     Utility for writing one or more columns to text file
     newline()
     write_header()
seisflows.plugins.line search.bracket module
class seisflows.plugins.line_search.bracket.Bracket(step_count_max=10,
                                                                    step\_len\_max=inf,
                                                                    path='/Users/niviyu/Documents/GitHub/seisflows/docs')
     Bases: seisflows.plugins.line search.base.Base
     Implements bracketing line search
     Variables x - list of step lenths from current line search f - correpsonding list of function values gtg - dot
          product of gradient with itself gtp - dot product of gradient and search direction
     Status codes status > 0: finished status == 0: not finished status < 0: failed
     calculate_step()
          Determines step length and search status
6.2.2 seisflows.plugins.optimize package
seisflows.plugins.optimize.LBFGS module
class seisflows.plugins.optimize.LBFGS.LBFGS(path='.',
                                                                       load=<function
                                                                                        loadnpy>,
                                                           save=<function savenpy>,
                                                                                       memory=5,
                                                           thresh=0.0, maxiter=inf, precond=None)
     Bases: object
     Limited-memory BFGS algorithm
     Includes optional safeguards: periodic restarting and descent conditions.
     To conserve memory, most vectors are read from disk rather than passed from a calling routine.
     apply (q, S=[], Y=[])
          Applies L-BFGS inverse Hessian to given vector
     check\_status(g, r)
     restart()
          Discards history and resets counters
```

```
update()
         Updates L-BFGS algorithm history
seisflows.plugins.optimize.LCG module
class seisflows.plugins.optimize.LCG.LCG(path, load=<function loadnpy>, save=<function
                                                 savenpy>,
                                                            thresh=inf,
                                                                        maxiter=inf,
                                                 cond=None)
    Bases: object
    CG solver
    apply_precond(r)
    check_status(*args, **kwargs)
    initialize()
    update (ap)
seisflows.plugins.optimize.NLCG module
class seisflows.plugins.optimize.NLCG.NLCG(path='.',
                                                               load=<function
                                                                                loadnpy>,
                                                                                thresh=1.0,
                                                   save=<function
                                                                   savenpy>,
                                                   maxiter=inf, precond=None)
    Nonlinear conjugate gradient method
    restart()
         Restarts algorithm
seisflows.plugins.optimize.NLCG.check_conjugacy(g_new, g_old)
seisflows.plugins.optimize.NLCG.check_descent(p_new, g_new)
seisflows.pluqins.optimize.NLCG.fletcher_reeves(g_new, g_old, precond=<function
                                                          <lambda>>)
                                                                g\_old,
seisflows.plugins.optimize.NLCG.pollak_ribere(g_new,
                                                                         precond=<function</pre>
                                                       <lambda>>)
seisflows.plugins.optimize.PLCG module
class seisflows.plugins.optimize.PLCG.LBFGS_(path='.',
                                                                                loadnpy>,
                                                                load=<function
                                                      save=<function savenpy>,
                                                                               memory=5,
                                                      thresh=0.0, maxiter=inf, precond=None)
    Bases: seisflows.plugins.optimize.LBFGS.LBFGS
    Adapts L-BFGS from nonlinear optimization to preconditioning
class seisflows.plugins.optimize.PLCG.PLCG(path, eta=1.0, **kwargs)
    Bases: seisflows.plugins.optimize.LCG.LCG
    Preconditioned truncated-Newton CG solver
    Adds preconditioning and adaptive stopping to LCG base class
    apply\_precond(r)
    check status(ap, verbose=True)
         Checks Eisenstat-Walker termination status
```

## 6.2.3 seisflows.plugins.preconds package

#### seisflows.plugins.preconds.diagonal module

```
class seisflows.plugins.preconds.diagonal.Diagonal
    Bases: object
    User supplied diagonal preconditioner
```

Rescales model parameters based on user supplied weights

## 6.2.4 seisflows.plugins.solver package

#### seisflows.plugins.solver.specfem2d module

```
seisflows.plugins.solver.specfem2d.smooth_legacy(input_path=", output_path=", parameters=[], span=0.0)

seisflows.plugins.solver.specfem2d.write_receivers(coords, path='.')

Writes receiver information to text file

seisflows.plugins.solver.specfem2d.write_sources(coords, path='.', ws=1.0, suffix=")

Writes source information to text file TODO this has to be adapted for new versions of specfem because the source file format has changed
```

## seisflows.plugins.solver.specfem3d module

```
seisflows.plugins.solver.specfem3d.write_receivers(h)
    Writes receiver information to text file
seisflows.plugins.solver.specfem3d.write_sources(PAR, h, path='.')
    Writes source information to text file
```

#### seisflows.plugins.solver.specfem3d\_globe module

```
seisflows.plugins.solver.specfem3d_globe.write_parameters (par, version)
    Writes parameters to text file
seisflows.plugins.solver.specfem3d_globe.write_receivers (h)
    Writes receiver information to text file
seisflows.plugins.solver.specfem3d_globe.write_sources (PAR, h, path='.')
    Writes source information to text file
```

#### 6.2.5 seisflows.plugins.solver\_io package

#### seisflows.plugins.solver io.adios module

```
seisflows.plugins.solver_io.adios.mread (path, parameters, iproc, prefix=", suffix=")
Multiparameter read, callable by a single mpi process
seisflows.plugins.solver_io.adios.read(path, parameter, iproc)
Reads from ADIOS container
```

```
seisflows.plugins.solver_io.adios.write(v, path, parameter, iproc)
Writes to ADIOS container
```

#### seisflows.plugins.solver\_io.fortran\_binary module

```
seisflows.plugins.solver_io.fortran_binary.copy_slice(src, dst, iproc, parameter)
    Copies SPECFEM model slice

seisflows.plugins.solver_io.fortran_binary.read_slice(path, parameters, iproc)
    Reads SPECFEM model slice(s) Such as, for example: proc000005_vp.bin In that specific case it would be: read_slice(path, 'vp', 5)

seisflows.plugins.solver_io.fortran_binary.write_slice(data, path, parameters, iproc)

Writes SPECFEM model slice
```

## 6.2.6 seisflows.plugins.adjoint module

```
seisflows.plugins.adjoint.Acceleration (syn, obs, nt, dt)
seisflows.plugins.adjoint.Amplitude(syn, obs, nt, dt)
     Cross correlation amplitude
seisflows.plugins.adjoint.Displacement(syn, obs, nt, dt)
seisflows.plugins.adjoint.Envelope(syn, obs, nt, dt, eps=0.05)
     Envelope difference (Yuan et al 2015, eq 16)
seisflows.plugins.adjoint.Envelope2 (syn, obs, nt, dt, eps=0.0)
     Envelope amplitude ratio (Yuan et al 2015, eq B-2, B-3)
seisflows.plugins.adjoint.Envelope3 (syn, obs, nt, dt, eps=0.0)
     Envelope cross-correlation lag (Yuan et al 2015, eqs B-2, B-5)
seisflows.plugins.adjoint.InstantaneousPhase (syn, obs, nt, dt, eps=0.05)
     Instantaneous phase (from Bozdag et al. 2011, eq 27)
seisflows.plugins.adjoint.InstantaneousPhase2(syn, obs, nt, dt, eps=0.0)
seisflows.plugins.adjoint.Traveltime (syn, obs, nt, dt)
     Cross correlation traveltime (Tromp et al 2005, eq 45)
seisflows.plugins.adjoint.TraveltimeInexact (syn, obs, nt, dt)
     Much faster (but possibly inaccurate) version of Traveltime function
seisflows.plugins.adjoint.Velocity (syn, obs, nt, dt)
seisflows.plugins.adjoint. Waveform (syn, obs, nt, dt)
     Waveform difference (Tromp et al 2005, eq 9)
```

#### 6.2.7 seisflows.plugins.misfit module

```
seisflows.plugins.misfit.Envelope (syn, obs, nt, dt, eps=0.05)
     Envelope difference (Yuan et al 2015, eq 9)
seisflows.plugins.misfit.Envelope2 (syn, obs, nt, dt, eps=0.0)
     Envelope amplitude ratio (Yuan et al 2015, eq B-1)
seisflows.plugins.misfit.Envelope3 (syn, obs, nt, dt, eps=0.0)
     Envelope cross-correlation lag (Yuan et al 2015, eqs B-4)
seisflows.plugins.misfit.InstantaneousPhase(syn, obs, nt, dt, eps=0.05)
     Instantaneous phase from Bozdag et al. 2011
seisflows.plugins.misfit.InstantaneousPhase2(syn, obs, nt, dt, eps=0.0)
seisflows.plugins.misfit.Traveltime (syn, obs, nt, dt)
     Compute cross correlation traveltime between two traces suposing that they contain only one arrival
seisflows.plugins.misfit.TraveltimeInexact (syn, obs, nt, dt)
     Much faster (but possibly inaccurate) version of Traveltime function
seisflows.plugins.misfit.Velocity (syn, obs, nt, dt)
seisflows.plugins.misfit.Waveform (syn, obs, nt, dt)
     Waveform difference
```

#### 6.2.8 seisflows.plugins.readers module

```
seisflows.plugins.readers.ascii (path, filenames)
    Reads SPECFEM3D-style ascii data
seisflows.plugins.readers.readBigSuFile (nameOfFile, nt, format='SU', byteorder='<')</pre>
```

This function is a hack to read .su file containing too many samples per traces. In the su format only 2 bytes per trace are dedicated to encoding for the number of samples (as signed int, see: http://lists.swapbytes.de/archives/obspy-users/2017-March/002359.html). Even if it's an old format it's still extremely stupid. This prove the lack of vision the designer of this format had at that time. They could have chosen 8 bytes or 16 bytes it was no big deal... They've cost me a day's work. But let us forget about the past. This limits the size of the traces to 32768 samples (NSTEP beween -32768 to 32768). Let us now suppose that we have NSTEP = 80000 samples per trace. We still want to use Obspy. The problem is that the NSTEP written in the .su file does not make any sense anymore and it is read by the obspy.read function! We thus rewrote a quick version of this function replacing the number of point by PAR.NT It is mainly copy-pasted from Obspy source code.

```
seisflows.plugins.readers.su(path, filename)
Reads Seismic Unix files Hardwired "
```

Function readBigSuFile is a hack to read su file containing too many samples per trace. In the su format only 2 bytes per trace are dedicated to encoding for the number of samples (as signed int, see: http://lists.swapbytes.de/archives/obspy-users/2017-March/002359.html). Even if it's an old format it's still extremely stupid. This proove the lack of vision the designer of this format had at that time. They could have chosen 8 bytes or 16 bytes it was no big deal... They've cost me a day's work. But let us forget about the past. This limits the size of the traces to 32768 samples (NSTEP beween -32768 to 32768). Let us now suppose that we have NSTEP = 40000 samples per trace. We still want to use Obspy. The problem is that the NSTEP written in the .su file does not make any sense anymore and it is read by the obspy.read function! We thus rewrote a quick version of this function from Obspy replacing the number of point by PAR.NT

#### 6.2.9 seisflows.plugins.wavelets module

```
seisflows.plugins.wavelets.gabor (nt, df, fp)
```

```
seisflows.plugins.wavelets.ricker (nt, dt, fp)
```

## 6.2.10 seisflows.plugins.writers module

```
seisflows.plugins.writers.ascii (stream, path, filenames)
Write ascii signal file
```

```
seisflows.plugins.writers.su(stream, path, filename)
```

Write Seismic Unix files. Function writeBigSuFile is a hack to write a .su file when the number of samples per trace is two big. In the su format only 2 bytes per trace are dedicated to encoding for the number of samples (as signed int, see: http://lists.swapbytes.de/archives/obspy-users/2017-March/002359.html). Even if it's an old format it's still extremely stupid. This proove the lack of vision the designer of this format had at that time. They could have chosen 8 bytes or 16 bytes it was no big deal... They've cost me a day's work. But let us forget about the past. This limits the size of the traces in the header to maximum 32768. We use Obspy to write the file with dummy values there instead of the real number of sample (that we now anyway: it is PAR.NT). We thus rewrote a quick version of this function from Obspy replacing the number of point by PAR.NT

```
seisflows.plugins.writers.writeBigSuFile(stream, path, byteorder='<')
```

This function is a hack to write a .su file when the number of samples per trace is two big. In the su format only 2 bytes per trace are dedicated to encoding for the number of samples (as signed int, see: http://lists.swapbytes.de/archives/obspy-users/2017-March/002359.html). Even if it's an old format it's still extremely stupid. This proove the lack of vision the designer of this format had at that time. They could have chosen 8 bytes or 16 bytes it was no big deal... They've cost me a day's work. But let us forget about the past. This limits the size of the traces in the header to maximum 32768. We use Obspy to write the file with dummy values there instead of the real number of sample (that we now anyway: it is PAR.NT). We thus rewrote a quick version of this function from Obspy replacing the number of point by PAR.NT This is mostly copy-pastes from Obspy source code

# 6.3 seisflows.postprocess package

# 6.3.1 seisflows.postprocess.base module

```
class seisflows.postprocess.base.base
Bases: object
Regularization, smoothing, sharpening, masking and related operations on models or gradients
check()
```

Checks parameters and paths

process\_kernels (path, parameters)

Sums kernels from individual sources, with optional smoothing

**Input path** directory containing sensitivity kernels

**Input parameters** list of material parameters e.g. ['vp','vs']

setup()

Placeholder for initialization or setup tasks

```
write gradient(path)
```

Combines contributions from individual sources and material parameters to get the gradient, and optionally applies user-supplied scaling

**Input path** directory from which kernels are read and to which gradient is written

#### 6.3.2 seisflows.postprocess.default module

```
class seisflows.postprocess.default.default
    Bases: seisflows.postprocess.base.base
    Default postprocesing option
```

Provides default image processing and regularization functions for models or gradients

# 6.4 seisflows.preprocess package

#### 6.4.1 seisflows.preprocess.base module

```
class seisflows.preprocess.base.base
     Bases: object
     Data preprocessing class
     Provides data processing functions for seismic traces, with options for data misfit, filtering, normalization and
     apply_csg_mute(traces)
     apply_filter(traces)
     apply_filter_backwards (traces)
     apply_mute(traces)
     apply_normalize(traces)
     check()
         Checks parameters and paths
     check filter()
         Checks filter settings
     check_mute()
         Checks mute settings
     check_normalize()
     csg_mute (seismo_w, T, T1)
     get_network_size(traces)
     get_receiver_coords (traces)
```

FIXME: extract time scheme from trace headers rather than parameters file. Note from Alexis Bottero: it is actually better like this in my opinion because this allows for longer traces to be processed. Indeed, in su format only 2 bytes are dedicated to the number of samples which is supposed to be stored as an unsigned int. The maximum NT which can be stored in the header is then 32762 whereas there is no limit in principle.

```
prepare_eval_grad (path='.')
```

get\_source\_coords (traces)
get\_time\_scheme (traces)

Prepares solver for gradient evaluation by writing residuals and adjoint traces

Input path directory containing observed and synthetic seismic data

```
setup()
     Sets up data preprocessing machinery
sum residuals(files)
     Sums squares of residuals
         Input files list of single-column text files containing residuals
         Output total_misfit sum of squares of residuals
write_adjoint_traces (path, syn, obs, channel)
     Writes "adjoint traces" required for gradient computation
         Input path location "adjoint traces" will be written
         Input syn obspy Stream object containing synthetic data
         Input obs obspy Stream object containing observed data
         Input channel channel or component code used by writer
write_residuals (path, syn, obs)
     Computes residuals
         Input path location "adjoint traces" will be written
         Input syn obspy Stream object containing synthetic data
         Input obs obspy Stream object containing observed data
```

#### 6.4.2 seisflows.preprocess.default module

```
class seisflows.preprocess.default.default
    Bases: seisflows.preprocess.base.base
```

Default preprocesing class

Provides data processing functions for seismic traces, with options for data misfit, filtering, normalization and muting

## 6.4.3 seisflows.preprocess.double difference module

```
class seisflows.preprocess.double_difference.double_difference
Bases: seisflows.preprocess.base.base
Double-difference data processing class
Adds double-difference data misfit functions to base class
adjoint_dd(si, sj, t0, nt, dt)
    Returns contribution to adjoint source from a single double difference measurement
apply_weights(traces)
check()
    Checks parameters, paths, and dependencies
distance(x1, y1, x2, y2)
load_weights()
shift(v, it)
    Shifts time series a given number of steps
```

```
sum_residuals (files)
    Sums squares of residuals

write_adjoint_traces (path, syn, dat, channel)
    Computes adjoint traces from observed and synthetic traces

write_residuals (path, syn, dat)
    Computes residuals from observations and synthetics
```

# 6.5 seisflows.solver package

#### 6.5.1 seisflows.solver.base module

```
class seisflows.solver.base.base
    Bases: object
```

**Provides an interface through which solver simulations can be set up** and run and a parent class for SPECFEM2D, SPECFEM3D and SPECFEM3D\_GLOBE subclasses

This class supports only acoustic and isotropic elastic inversions. For additional options, see github.com/rmodrak/seisflows-multiparameter

eval\_func, eval\_grad, apply\_hess These methods deal with evaluation of the misfit function or its derivatives. Together, they provide the primary interface through which SeisFlows interacts with SPECFEM2D/3D

**forward, adjoint** These methods allow direct access to low-level SPECFEM2D/3D components, providing an alternative interface through which to interact with the solver

**setup, generate\_data, generate\_model** One-time operations performed at beginning of an inversion or migration

#### initialize solver directories, initialize adjoint traces

Checks parameters and paths

SPECFEM2D/3D requires a particular directory structure in which to run and particular file formats for models, data, and parameter files. These methods help put in place all these prerequisites

**load, save** For reading and writing SPECFEM2D/3D models and kernels. On the disk, models and kernels are stored as binary files, and in memory, as dictionaries with different keys corresponding to different material parameters

**split, merge** Within the solver routines, it is natural to store models as dictionaries. Within the optimization routines, it is natural to store models as vectors. Two methods, 'split' and 'merge', are used to convert back and forth between these two representations

combine, smooth Utilities for combining and smoothing kernels

```
adjoint()
    Calls adjoint solver
apply_hess(path=")
    Computes action of Hessian on a given model vector. (A gradient evaluation must have already been carried out.)
    Input path: directory to which output files are exported
check()
```

```
check mesh properties(path=None)
     path contains binary files such as:
                                           proc000000_z.bin, proc000000_x.bin, proc000000_vs.bin
     proc000000 vp.bin, proc000000 rho.bin, proc000001 z.bin, proc000001 x.bin, proc000001 vs.bin ...
     These will be read to get the number of processors used, the number of gll points and the coordinates of
     those points
check solver parameter files()
check source names()
     Determines names of sources by applying wildcard rule to user- supplied input files If source prefix
     is 'SOURCE' and that in specfem DATA folder are the files SOURCE_00001, SOURCE_00002,
     SOURCE_00003,... Then this will build the list names = ['00001','00002','00003',...] If, for ex, taskid
     is 1 the function returns ['00001', '00002']
clean()
combine (input_path=", output_path=", parameters=[])
     Sums individual source contributions. Wrapper over xcombine_sem utility.
cwd
data_filenames
eval_func (path=", export_traces=False, write_residuals=True)
     Performs forward simulations needed for misfit function evaluation
         Input path: directory from which model is imported
         Input export_traces: save or discard traces?
eval_grad (path=", export_traces=False)
     Evaluates gradient by carrying out adjoint simulations. (A function evaluation must already have been
     carried out.)
         Input path: directory from which model is imported
         Input export_traces: save or discard traces?
export_kernels(path)
export_model (path, parameters=['rho', 'vp', 'vs'])
export residuals(path)
export traces (path, prefix='traces/obs')
forward()
    Calls forward solver
generate data(*args, **kwargs)
     Generates data
generate_mesh (*args, **kwargs)
     Performs meshing and database generation
import_model (path)
import_traces (path)
initialize_adjoint_traces()
     Puts in place "adjoint traces" expected by SPECFEM Adjoint traces are initialized by writing zeros for
```

Puts in place "adjoint traces" expected by SPECFEM Adjoint traces are initialized by writing zeros for all channels. (even the ones that are not actually in use, as required by specfem) Ex: Ux, Uy, Uz even if only Uy is used Channels actually in use during an inversion or migration will be overwritten with nonzero values later on.

```
initialize solver directories()
     Creates directory structure expected by SPECFEM3D, copies executables, and prepares input files. Ex-
     ecutables must be supplied by user as there is currently no mechanism for automatically compiling from
     source.
io
     Solver IO module
kernel databases
load (path, parameters=[], prefix=", suffix=")
     Loads SPECFEM2D/3D models or kernels
         Input path: directory from which model is read
         Input parameters: list of material parameters to be read (if empty, defaults to self.parameters)
         Input prefix: optional filename prefix
         Input suffix: optional filename suffix, eg '_kernel'
         Output dict: model or kernels indexed by material parameter and processor rank, ie
             dict[parameter][iproc]
merge (model, parameters=[])
     Converts model from dictionary to vector representation
mesh_properties
model_databases
rename data(path)
     Works around conflicting data filename conventions
rename_kernels()
     Works around conflicting kernel filename conventions
save (dict, path, parameters=['vp', 'vs', 'rho'], prefix=", suffix=")
     Saves SPECFEM2D/3D models or kernels
         Input dict: model stored as a dictionary or Container
         Input path: directory to which model is written
         Input parameters: list of material parameters to be written
         Input prefix : optional filename prefix
         Input suffix: optional filename suffix, eg '_kernel'
setup()
     Prepares solver for inversion or migration Sets up directory structure expected by SPECFEM and copies
     or generates seismic data to be inverted or migrated
smooth (input_path=", output_path=", parameters=[], span=0.0)
     Smooths kernels by convolving them with a Gaussian. Wrapper over xsmooth_sem utility.
source_name
source names
source_prefix
split (m, parameters=[])
     Converts model from vector to dictionary representation
taskid
```

#### 6.5.2 seisflows.solver.specfem2d module

```
class seisflows.solver.specfem2d.specfem2d
     Bases: seisflows.solver.base.base
     Python interface for SPECFEM2D
     See base class for method descriptions
     adjoint()
          Calls SPECFEM2D adjoint solver
     check()
          Checks parameters and paths
     check_solver_parameter_files()
          Checks solver parameters
     data filenames
     export model (path)
     forward (path='traces/syn')
          Calls SPECFEM2D forward solver
     generate_data(**model_kwargs)
          Generates data (perform meshing and database generation first)
     generate_mesh (model_path=None, model_name=None, model_type='gll')
          Performs meshing and database generation
     import_model (path)
     initialize_adjoint_traces()
          Puts in place "adjoint traces" expected by SPECFEM Adjoint traces are initialized by writing zeros for
          all channels. (even the ones that are not actually in use, as required by specfem) Ex: Ux, Uy, Uz even if
          only Uy is used Channels actually in use during an inversion or migration will be overwritten with nonzero
          values later on.
     kernel_databases
     model databases
     source prefix
```

## 6.5.3 seisflows.solver.specfem3d module

```
class seisflows.solver.specfem3d.specfem3d
    Bases: seisflows.solver.base.base
    Python interface for SPECFEM3D
    See base class for method descriptions
    adjoint()
        Calls SPECFEM3D adjoint solver
    check()
        Checks parameters and paths
    check_solver_parameter_files()
        Checks solver parameters
```

```
data filenames
data wildcard
eval_func(*args, **kwargs)
     Performs forward simulations needed for misfit function evaluation
         Input path: directory from which model is imported
         Input export_traces: save or discard traces?
forward (path='traces/syn')
     Calls SPECFEM3D forward solver
generate_data(**model_kwargs)
     Generates data
generate_mesh (model_path=None, model_name=None, model_type='gll')
     Performs meshing and database generation
initialize_adjoint_traces()
     Puts in place "adjoint traces" expected by SPECFEM Adjoint traces are initialized by writing zeros for
     all channels. (even the ones that are not actually in use, as required by specfem) Ex: Ux, Uy, Uz even if
     only Uy is used Channels actually in use during an inversion or migration will be overwritten with nonzero
     values later on.
kernel_databases
model databases
rename data()
     Works around conflicting data filename conventions
source_prefix
write_parameters()
write_receivers()
write_sources()
```

#### 6.5.4 seisflows.solver.specfem3d globe module

# 6.6 seisflows.system package

#### 6.6.1 seisflows.system.base module

```
class seisflows.system.base.base
Bases: object
Abstract base class
check()
        Checks parameters and paths
checkpoint (path, classname, method, args, kwargs)
        Writes information to disk so tasks can be executed remotely
run (classname, method, *args, **kwargs)
        Runs task multiple times
```

```
run_single (classname, method, *args, **kwargs)
    Runs task a single time
submit ()
    Submits workflow
taskid()
    Provides a unique identifier for each running task
```

#### 6.6.2 seisflows.system.lsf lg module

```
class seisflows.system.lsf_lg.lsf_lg
    Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

Intermediate files are written to a global scratch path PATH.SCRATCH, which must be accessible to all compute nodes.

Optionally, users can provide a local scratch path PATH.LOCAL if each compute node has its own local filesystem.

For important additional information, please see http://seisflows.readthedocs.org/en/latest/manual.html# system-configuration

```
check()
     Checks parameters and paths
job_id_list(stdout)
job_status (classname, method, jobs)
mpiexec()
     Specifies MPI executable used to invoke solver
run (classname, method, hosts='all', **kwargs)
     Runs task multiple times in embarrassingly parallel fasion
     Executes classname.method(*args, **kwargs) NTASK times, each time on NPROC cpu cores
run_single (classname, method, hosts='all', **kwargs)
     Runs task multiple times in embarrassingly parallel fasion
     Executes classname.method(*args, **kwargs) NTASK times, each time on NPROC cpu cores
save_kwargs (classname, method, kwargs)
submit (workflow)
     Submits workflow
taskid()
    Provides a unique identifier for each running task
timestamp()
```

#### 6.6.3 seisflows.system.lsf sm module

#### 6.6.4 seisflows.system.multicore module

```
class seisflows.system.multicore.multicore
    Bases: seisflows.system.serial.serial
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

For important additional information, please see <a href="http://seisflows.readthedocs.org/en/latest/manual.html#">http://seisflows.readthedocs.org/en/latest/manual.html#</a> system-configuration

```
check()
     Checks parameters and paths

run (classname, method, *args, **kwargs)
     Runs task multiple times in embarrassingly parallel fasion
     Executes classname.method(*args, **kwargs) NTASK times, each time on NPROC cpu cores

run_single (classname, method, *args, **kwargs)
     Runs task a single time

save_kwargs (classname, method, kwargs)

submit (workflow)
     Submits job
```

#### 6.6.5 seisflows.system.multithreaded module

```
class seisflows.system.multithreaded.multithreaded
    Bases: seisflows.system.multicore.multicore
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

For important additional information, please see <a href="http://seisflows.readthedocs.org/en/latest/manual.html#">http://seisflows.readthedocs.org/en/latest/manual.html#</a> system-configuration

```
check ()

Checks parameters and paths
```

## 6.6.6 seisflows.system.pbs\_lg module

```
class seisflows.system.pbs_lg.pbs_lg
    Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

Intermediate files are written to a global scratch path PATH.SCRATCH, which must be accessible to all compute nodes.

Optionally, users can provide a local scratch path PATH.LOCAL if each compute node has its own local filesystem.

For important additional information, please see http://seisflows.readthedocs.org/en/latest/manual.html# system-configuration

```
check()
     Checks parameters and paths
job_array_args (hosts)
job_array_cmd (classname, method, hosts)
job_array_status (classname, method, jobs)
     Determines completion status of one or more jobs
mpiexec()
     Specifies MPI executable used to invoke solver
run (classname, method, hosts='all', **kwargs)
     Runs task multiple times in embarrassingly parallel fasion
     Executes classname.method(*args, **kwargs) NTASK times, each time on NPROC cpu cores
save_kwargs (classname, method, kwargs)
submit (workflow)
     Submits workflow
submit_job_array (classname, method, hosts='all')
taskid()
```

## 6.6.7 seisflows.system.pbs sm module

Provides a unique identifier for each running task

```
class seisflows.system.pbs_sm.pbs_lg
    Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

Intermediate files are written to a global scratch path PATH.SCRATCH, which must be accessible to all compute nodes.

Optionally, users can provide a local scratch path PATH.LOCAL if each compute node has its own local filesystem.

For important additional information, please see <a href="http://seisflows.readthedocs.org/en/latest/manual/manual.html#">http://seisflows.readthedocs.org/en/latest/manual/manual.html#</a> system-configuration

```
check()
    Checks parameters and paths
hostlist()
    Generates list of allocated cores
```

```
mpiexec()
    Specifies MPI executable used to invoke solver

run (classname, method, hosts='all', **kwargs)
    Runs embarrassingly parallel tasks

Executes the following multiple times: classname.method(*args, **kwargs)
    system.taskid serves to provide each running task a unique identifier

save_kwargs (classname, method, kwargs)

submit (workflow)
    Submits workflow

taskid()
    Provides a unique identifier for each running task
```

# 6.6.8 seisflows.system.serial module

```
class seisflows.system.serial.serial
    Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

For important additional information, please see <a href="http://seisflows.readthedocs.org/en/latest/manual/manual.html#">http://seisflows.readthedocs.org/en/latest/manual/manual.html#</a> system-configuration

```
check()
```

Checks parameters and paths

```
mpiexec()
```

Specifies MPI executable used to invoke solver

```
progress(taskid)
```

Provides status update

```
run (classname, method, hosts='all', **kwargs)
```

Executes method from classname multiple times in serial taskid is used to identified a given task (one source)

```
run_single (classname, method, *args, **kwargs)
```

Runs task a single time

```
submit (workflow)
```

Submits job

### taskid()

Return the value of the environment variable SEISFLOWS\_TASKID which provides a unique identifier for each running task

# 6.6.9 seisflows.system.slurm Ig module

```
class seisflows.system.slurm_lg.slurm_lg
    Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

Intermediate files are written to a global scratch path PATH.SCRATCH, which must be accessible to all compute nodes.

Optionally, users can provide a local scratch path PATH.LOCAL if each compute node has its own local filesystem.

For important additional information, please see http://seisflows.readthedocs.org/en/latest/manual.html# system-configuration

### check()

Checks parameters and paths

### job\_array\_status (classname, method, jobs)

Determines completion status of job or job array

### job id list(stdout, ntask)

Parses job id list from sbatch standard output

### job\_status(job)

Queries completion status of a single job

#### mpiexec()

Specifies MPI executable used to invoke solver

```
run (classname, method, *args, **kwargs)
```

Runs task multiple times in embarrassingly parallel fasion

Executes classname.method(\*args, \*\*kwargs) NTASK times, each time on NPROC cpu cores

```
run_single (classname, method, *args, **kwargs)
```

Runs task a single time

Executes classname.method(\*args, \*\*kwargs) a single time on NPROC cpu cores

### submit (workflow)

Submits workflow

### taskid()

Provides a unique identifier for each running task

# 6.6.10 seisflows.system.slurm sm module

```
class seisflows.system.slurm_sm.slurm_sm
Bases: seisflows.system.base.base
```

An interface through which to submit workflows, run tasks in serial or parallel, and perform other system functions.

By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

Intermediate files are written to a global scratch path PATH.SCRATCH, which must be accessible to all compute nodes.

Optionally, users can provide a local scratch path PATH.LOCAL if each compute node has its own local filesystem.

For important additional information, please see http://seisflows.readthedocs.org/en/latest/manual.html# system-configuration

# 6.6.11 seisflows.system.tiger\_lg module

```
class seisflows.system.tiger_lg.tiger_lg
   Bases: seisflows.system.slurm_lg.slurm_lg
   Specially designed system interface for tiger.princeton.edu
   See parent class for more information.
   check()
        Checks parameters and paths
   submit(*args, **kwargs)
        Submits job
```

## 6.6.12 seisflows.system.tiger sm module

```
class seisflows.system.tiger_sm
   Bases: seisflows.system.slurm_sm.slurm_sm
   Specially designed system interface for tiger.princeton.edu
   See parent class for more information.
   check()
        Checks parameters and paths
   submit(*args, **kwargs)
        Submits job
```

# 6.7 seisflows.tools package

# 6.7.1 seisflows.tools.array module

```
seisflows.tools.array.count_zeros(a)
     Counts number of zeros in a list or array
seisflows.tools.array.grid2mesh(V, grid, mesh)
     Interpolates from structured coordinates (grid) to unstructured coordinates (mesh)
seisflows.tools.array.gridsmooth(Z, span)
     Smooths values on 2D rectangular grid
seisflows.tools.array.loadnpy (filename)
     Loads numpy binary file.
seisflows.tools.array.mesh2grid(v, mesh)
     Interpolates from an unstructured coordinates (mesh) to a structured coordinates (grid)
seisflows.tools.array.meshsmooth(v, mesh, span)
     Smooths values on 2D unstructured mesh
seisflows.tools.array.savenpy(filename, v)
     Saves numpy binary file.
seisflows.tools.array.sortrows(a, return_index=False, return_inverse=False)
     Sorts rows of numpy array
seisflows.tools.array.stack(*args)
seisflows.tools.array.uniquerows(a, sort_array=False, return_index=False)
     Finds unique rows of numpy array
```

### 6.7.2 seisflows.tools.err module

```
exception seisflows.tools.err.ParameterError(*args)
Bases: exceptions.ValueError
```

# 6.7.3 seisflows.tools.graphics module

```
seisflows.tools.graphics.get_regular_ticks(v, interval)
Returns regular tick intervals.

seisflows.tools.graphics.plot_gll(x, y, z, vmin=None, vmax=None)
Plots values on 2D unstructured GLL mesh

seisflows.tools.graphics.plot_many_gll(x, y, z, vmin=None, vmax=None)
Plots values on big 2D unstructured GLL mesh (in that case tricontourf does not work)

seisflows.tools.graphics.plot_section(stream, ax=None, cmap='seismic', clip=100, title=", x_interval=1.0, y_interval=1.0)
Plots a seismic section from an obspy stream.
```

### **Parameters**

- stream (Obspy stream object) Obspy stream object created from a SU data file
- ax (Matplotlib Axes object) Optional axis object

- cmap (str) Matplotlib colormap option.
- clip (float) Percentage value (0-100) for amplitude clipping
- title (str) plot title
- **x\_interval** (float) Offset axis tick interval in km
- y\_interval (float) Time axis tick interval in km

Raises NotImplementedError - If stream object does not have SU format

seisflows.tools.graphics.plot\_vector(t, v, xlabel=", ylabel=", title=")
Plots a vector or time series.

### **Parameters**

- $\mathbf{v}$  (ndarray, ndims = 1/2) Vector or time series to plot
- xlabel (str) x axis label
- ylabel (str) y axis label
- title (str) plot title

Raises ValueError – If dimensions of v are greater than 2

## 6.7.4 seisflows.tools.math module

```
seisflows.tools.math.angle (x, y)
seisflows.tools.math.backtrack2 (f0, g0, x1, f1, b1=0.1, b2=0.5)
     Safeguarded parabolic backtrack
seisflows.tools.math.backtrack3 (f0, g0, x1, f1, x2, f2)
     Safeguarded cubic backtrack
seisflows.tools.math.dot (x, y)
seisflows.tools.math.gauss2(X, Y, mu, sigma, normalize=True)
     Evaluates Gaussian over points of X,Y
seisflows.tools.math.grad(V, h=[])
     Evaluates derivatives on a 2D rectangular grid
seisflows.tools.math.hilbert(w)
seisflows.tools.math.lsq2 (x, f)
     Parabolic least squares fit
seisflows.tools.math.nabla (V, h=I)
     Returns sum of first-order spatial derivatives of a function defined on a 2D rectangular grid; generalizes Lapla-
     cian
seisflows.tools.math.nabla2 (V, h=[])
     Returns sum of second-order spatial derivatives of a function defined on a 2D rectangular grid; generalizes
     Laplacian
seisflows.tools.math.polyfit2(x, f)
     Parabolic fit
seisflows.tools.math.tv(Z, h=[], epsilon=1e-06)
```

# 6.7.5 seisflows.tools.msg module

## 6.7.6 seisflows.tools.seismic module

```
class seisflows.tools.seismic.Container
     Bases: collections.defaultdict
     Dictionary-like object for holding models or kernels
class seisflows.tools.seismic.Minmax
     Bases: collections.defaultdict
     Keeps track of min, max values of model or kernel
     update ([E], **F) \rightarrow None. Update D from dict/iterable E and F.
          If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method,
          does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
class seisflows.tools.seismic.Writer(path='./output.stat')
     Bases: object
     Utility for appending values to text files
seisflows.tools.seismic.call_solver(mpiexec, executable, output='solver.log')
     Calls MPI solver executable
     A less complicated version, without error catching, would be subprocess.call(mpiexec +' '+ executable,
     shell=True)
seisflows.tools.seismic.getpar(key, file='DATA/Par_file', sep='=', cast=<type 'str'>, noOut-
                                         put=False)
     Reads parameter from text file
seisflows.tools.seismic.setpar(key, val, filename='DATA/Par_file', path='.', sep='=')
     Writes parameter to text file
```

# 6.7.7 seisflows.tools.signal module

```
seisflows.tools.signal.correlate(u, v)

seisflows.tools.signal.mask (slope, const, offset, time_scheme, length=400)
Constructs tapered mask that can be applied to trace to mute early or late arrivals.

seisflows.tools.signal.mute_early_arrivals (traces, slope, const, time_scheme, s_coords, r_coords)
Applies tapered mask to record section, muting early arrivals

Signals arriving before

SLOPE * || s - r || + CONST

are muted, where slope is has units of velocity**-1, CONST has units of time, and || s - r || is distance between source and receiver.

seisflows.tools.signal.mute_late_arrivals (traces, slope, const, time_scheme, s_coords, r_coords)

Applies tapered mask to record section, muting late arrivals

Signals arriving after

SLOPE * || s - r || + CONST
```

```
are muted, where SLOPE is has units of velocity**-1, CONST has units of time, and \| s - r \| is distance between source and receiver. 
seisflows.tools.signal.mute_long_offsets(traces, dist, s_coords, r_coords) Mutes traces having \| s - r \| > DIST
```

where  $\| s - r \|$  is the offset between source and receiver and DIST is a user-supplied cutoff seisflows.tools.signal.mute\_short\_offsets(traces, dist, s\_coords, r\_coords)

Mutes traces having

 $\parallel s - r \parallel < DIST$ 

where  $\parallel$  s - r  $\parallel$  is the offset between source and receiver and DIST is a user-supplied cutoff

```
seisflows.tools.signal.sconvolve (s, h, w, inplace=True) seisflows.tools.signal.tukeywin (nt, imin, imax, alpha=0.05)
```

## 6.7.8 seisflows.tools.tools module

```
class seisflows.tools.tools.Struct(*args, **kwargs)
    Bases: dict
seisflows.tools.tools.call(*args, **kwargs)
seisflows.tools.tools.diff(list1, list2)
    Difference between unique elements of lists
seisflows.tools.tools.divides (i, j)
    True if j divides i
seisflows.tools.tools.exists(names)
    Wrapper for os.path.exists
seisflows.tools.tools.findpath(name)
    Resolves absolute path of module
seisflows.tools.tools.getset(arg)
seisflows.tools.tools.iterable(arg)
seisflows.tools.tools.loadjson(filename)
    Load object using json
seisflows.tools.tools.loadnpy (filename)
    Loads numpy binary file.
seisflows.tools.tools.loadobj(filename)
    Load object using pickle
seisflows.tools.tools.loadpy (filename)
seisflows.tools.tools.loadtxt(filename)
    Load scalar from text file
seisflows.tools.tools.loadyaml (filename)
seisflows.tools.tools.module_exists(name)
seisflows.tools.nproc()
seisflows.tools.tools.package_exists(name)
```

```
seisflows.tools.tools.pkgpath(name)
seisflows.tools.tools.savejson(filename, obj)
    Save object using json
seisflows.tools.tools.savenpy(filename, v)
    Saves numpy binary file.
seisflows.tools.tools.saveobj(filename, obj)
    Save object using pickle
seisflows.tools.tools.savetxt(filename, v)
    Save scalar to text file
seisflows.tools.tools.timestamp()
```

### 6.7.9 seisflows.tools.unix module

```
seisflows.tools.unix.cat(src, *dst)
     Open a file and print it. If file dst is given the file is printed into dst
seisflows.tools.unix.cd(path)
     Change current directory to the one given
seisflows.tools.unix.cp(src=", dst=")
     Copy all (files or directories, as list or tupples) given in src to directory (or file) dst
seisflows.tools.unix.hostname()
seisflows.tools.unix.ln(src, dst)
seisflows.tools.unix.ls(path)
seisflows.tools.unix.mkdir(dirs)
seisflows.tools.unix.mv(src=", dst=")
seisflows.tools.unix.rename(old, new, names)
seisflows.tools.unix.rm(path=")
seisflows.tools.unix.select(items, prompt=")
seisflows.tools.unix.touch(filename, times=None)
seisflows.tools.unix.which(name)
```

# 6.8 seisflows.workflow package

## 6.8.1 seisflows.workflow.base module

```
class seisflows.workflow.base.base
   Bases: object
   Workflow abstract base class
   check()
        Checks parameters and paths
   checkpoint()
        Writes information to disk so workflow can be resumed following a break
```

```
main()
```

Main routine

Execution of a workflow is equivalent to stepping through workflow.main

## 6.8.2 seisflows.workflow.inversion module

```
class seisflows.workflow.inversion.inversion
    Bases: seisflows.workflow.base.base
```

Waveform inversion base class

Peforms iterative nonlinear inversion and provides a base class on top of which specialized strategies can be implemented.

To allow customization, the inversion workflow is divided into generic methods such as 'initialize', 'finalize', 'evaluate\_function', 'evaluate\_gradient', which can be easily overloaded.

Calls to forward and adjoint solvers are abstracted through the 'solver' interface so that various forward modeling packages can be used interchangeably.

Commands for running in serial or parallel on a workstation or cluster are abstracted through the 'system' interface.

```
check()
```

Checks parameters and paths

### checkpoint()

Writes information to disk so workflow can be resumed following a break

#### clean()

Cleans directories in which function and gradient evaluations were carried out

### compute\_direction()

Computes search direction

### evaluate\_function()

Performs forward simulation to evaluate objective function

### evaluate\_gradient()

Performs adjoint simulation to evaluate gradient

#### finalize()

Saves results from current model update iteration

### initialize()

Prepares for next model update iteration

## line\_search()

Conducts line search in given search direction

**Status codes** status > 0: finished status == 0: not finished status < 0: failed

### main()

Carries out seismic inversion

```
save_gradient()
save_kernels()
save_model()
```

save\_residuals()

```
save_traces()
     setup()
          Lays groundwork for inversion. Runs setup method for preprocess, postprocess, optimize and also for
          the solver through the system: system.run('solver','setup') This will copy user supplied data or generate
          it from a 'real' model supplied This will setup the mesh for the initial model supplied If path LOCAL is
          supplied this is done even at iteration > 1
     write_gradient (path=", suffix=")
          Writes gradient in format expected by nonlinear optimization library
     write_misfit (path=", suffix=")
          Writes misfit in format expected by nonlinear optimization library
     write_model (path=", suffix=")
          Writes model in format expected by solver
6.8.3 seisflows.workflow.migration module
class seisflows.workflow.migration.migration
     Bases: seisflows.workflow.base.base
     Migration base class.
     In the terminology of seismic exploration, implements a 'reverse time migration'.
     check()
         Checks parameters and paths
     main()
          Migrates seismic data
     prepare_model()
     save_kernels()
     save_kernels_sum()
     save_traces()
6.8.4 seisflows.workflow.test_adjoint module
seisflows.workflow.test adjoint.DotProductLHS(keys, x, y)
seisflows.workflow.test_adjoint.DotProductRHS(keys, x, y)
class seisflows.workflow.test_adjoint.test_adjoint
     Bases: seisflows.workflow.base.base
```

# 6.8. seisflows.workflow package

Main routine

prepare\_model()

Checks parameters and paths

Execution of a workflow is equivalent to stepping through workflow.main

check()

event main()

## 6.8.5 seisflows.workflow.test forward module

```
class seisflows.workflow.test_forward.test_forward
    Bases: seisflows.workflow.base.base
    Tests solver by running forward simulation
    check()
        Checks parameters and paths
    main()
        Generates seismic data
```

# 6.8.6 seisflows.workflow.test\_optimize module

## 6.8.7 seisflows.workflow.test\_postprocess module

# 6.8.8 seisflows.workflow.test\_preprocess module

```
class seisflows.workflow.test_preprocess.test_preprocess
    Bases: seisflows.workflow.base.base
    Signal processing integration test
    check()
        Checks parameters and paths
main()
        Tests data processing methods
save(data, filename)
test_adjoint(dat, syn)
test_filter(dat)
test_misfit(dat, syn)
test_mute(dat)
test_normalize(dat)
test_reader()
test_writer(data)
```

# 6.8.9 seisflows.workflow.test system module

```
class seisflows.workflow.test_system.test_system
   Bases: seisflows.workflow.base.base
   Tests system interface
   check()
        Checks parameters and paths
   hello (msg='Hello from %d')
        Prints hello message
   main()
        Main routine
        Execution of a workflow is equivalent to stepping through workflow.main
```

# 6.8.10 seisflows.workflow.thrifty\_inversion module

```
class seisflows.workflow.thrifty_inversion.thrifty_inversion
    Bases: seisflows.workflow.inversion.inversion
    Thrifty inversion subclass
    Provides savings over conventional inversion by carrying over forward simulations from line search
    The results of 'inversion' and 'thrifty_inversion' should be exactly the same
    clean()
        Cleans directories in which function and gradient evaluations were carried out
    initialize()
        Prepares for next model update iteration
    status = 0
    update_status()
```

# 6.9 seisflows.config module

```
class seisflows.config.Dict (newdict)
    Bases: object
    Dictionary-like object for holding parameters or paths
    update (newdict)

class seisflows.config.Null(*args, **kwargs)
    Bases: object
    Always and reliably does nothing

seisflows.config.config()
    Instantiates SeisFlows objects and makes them globally accessible by registering them in sys.modules

seisflows.config.custom_import(*args)
    Imports SeisFlows module and extracts class of same name. For example,
    custom_import('workflow', 'inversion')
```

imports 'seisflows.workflow.inversion' and, from this module, extracts class 'inversion'.

 $\verb"seisflows.config.load" (path")$ 

Imports session from disk

seisflows.config.save()

Exports session to disk Write files: seisflows\_parameters.json, seisflows\_paths.json, seisflows\_system.p, seisflows\_preprocess.p, seisflows\_postprocess.p, seisflows\_optimize.p, seisflows\_workflow.p

seisflows.config.tilde\_expand(mydict)

Expands tilde character in path strings

# **PYTHON MODULE INDEX**

```
S
                                           seisflows.system.pbs lq,32
                                           seisflows.system.pbs_sm, 33
seisflows.config,45
                                           seisflows.system.serial, 34
seisflows.optimize.base, 15
                                           seisflows.system.slurm lq,34
seisflows.optimize.LBFGS, 16
                                          seisflows.system.slurm sm, 35
seisflows.optimize.NLCG, 16
                                          seisflows.system.tiger_lg, 36
seisflows.optimize.steepest_descent, 17
                                           seisflows.system.tiger sm, 36
seisflows.plugins.adjoint, 21
seisflows.plugins.line_search.backtrack, seisflows.tools.array, 37
                                           seisflows.tools.err, 37
                                          seisflows.tools.graphics, 37
seisflows.plugins.line_search.base, 17
                                           seisflows.tools.math, 38
seisflows.plugins.line_search.bracket,
                                           seisflows.tools.msg, 39
                                           seisflows.tools.seismic, 39
seisflows.plugins.misfit, 21
                                           seisflows.tools.signal, 39
seisflows.plugins.optimize.LBFGS, 18
                                           seisflows.tools.tools,40
seisflows.plugins.optimize.LCG, 19
                                           seisflows.tools.unix,41
seisflows.plugins.optimize.NLCG, 19
                                           seisflows.workflow.base,41
seisflows.plugins.optimize.PLCG, 19
                                           seisflows.workflow.inversion, 42
seisflows.plugins.preconds.diagonal, 20
                                           seisflows.workflow.migration, 43
seisflows.plugins.readers, 22
                                           seisflows.workflow.test adjoint, 43
seisflows.plugins.solver.specfem2d, 20
                                           seisflows.workflow.test forward,44
seisflows.plugins.solver.specfem3d, 20
seisflows.plugins.solver.specfem3d_globe, seisflows.workflow.test_postprocess, 44
                                           seisflows.workflow.test_preprocess,44
                                           seisflows.workflow.test_system, 45
seisflows.plugins.solver_io.adios, 20
\verb|seisflows.plugins.solver_io.fortran_binary, is flows.workflow.thrifty\_inversion, \\
seisflows.plugins.wavelets, 22
seisflows.plugins.writers, 23
seisflows.postprocess.base, 23
seisflows.postprocess.default, 24
seisflows.preprocess.base, 24
seisflows.preprocess.default, 25
seisflows.preprocess.double difference,
       2.5
seisflows.solver.base, 26
seisflows.solver.specfem2d,29
seisflows.solver.specfem3d,29
seisflows.system.base, 30
seisflows.system.lsf_lq,31
seisflows.system.lsf_sm, 32
seisflows.system.multicore, 32
seisflows.system.multithreaded, 32
```

48 Python Module Index

# **INDEX**

A	В
Acceleration() (in module seis-	Backtrack (class in seis-
flows.plugins.adjoint), 21	flows.plugins.line_search.backtrack), 17
Acceleration() (in module seisflows.plugins.misfit),	backtrack2() (in module seisflows.tools.math), 38
21	backtrack3() (in module seisflows.tools.math), 38
adjoint() (seisflows.solver.base.base method), 26	base (class in seisflows.optimize.base), 15
$\verb"adjoint()" (se is flows. solver. spec fem 2d. spec fem 2d.$	Base (class in seisflows.plugins.line_search.base), 17
method), 29	base (class in seisflows.postprocess.base), 23
$\verb"adjoint()" (se is flows. solver. spec fem 3d. spec fem 3d$	
method), 29	base (class in seisflows.solver.base), 26
adjoint_dd() (seis-	base (class in seisflows.system.base), 30
flows.preprocess.double_difference.double_diffe	re <b>nce</b> s ∈ (class in seisflows.workflow.base), 41
method), 25	Bracket (class in seis-
Amplitude() (in module seisflows.plugins.adjoint), 21	$flows.plugins.line\_search.bracket), 18$
Amplitude() (in module seisflows.plugins.misfit), 21	
angle() (in module seisflows.tools.math), 38	C
apply() (seisflows.plugins.optimize.LBFGS.LBFGS	calculate_step() (seis-
method), 18	flows.plugins.line_search.backtrack.Backtrack
<pre>apply_csg_mute() (seisflows.preprocess.base.base</pre>	method), 17
method), 24	calculate_step() (seis-
<pre>apply_filter() (seisflows.preprocess.base.base</pre>	flows.plugins.line_search.base.Base method),
method), 24	17
apply_filter_backwards() (seis-	calculate_step() (seis-
flows.preprocess.base.base method), 24	flows.plugins.line_search.bracket.Bracket
$\verb"apply_hess"()" (\textit{seisflows.solver.base.base method}), 26$	method), 18
apply_mute() (seisflows.preprocess.base.base	call () (in module seisflows.tools.tools), 40
method), 24	<pre>call_solver() (in module seisflows.tools.seismic),</pre>
<pre>apply_normalize() (seisflows.preprocess.base.base</pre>	39
method), 24	cat () (in module seisflows.tools.unix), 41
apply_precond() (seis-	cd() (in module seisflows.tools.unix), 41
flows.plugins.optimize.LCG.LCG method),	check () (seisflows.optimize.base.base method), 15
19	check() (seisflows.optimize.LBFGS.LBFGS method),
apply_precond() (seis-	16
flows.plugins.optimize.PLCG.PLCG method),	check () (seisflows.optimize.NLCG.NLCG method), 16
19	check() (seisflows.optimize.steepest_descent.steepest_descent
apply_weights() (seis-	method), 17
flows.preprocess.double_difference.double_diffe	reassection (seisflows.postprocess.base.base method), 23
method), 25	check() (seisflows.preprocess.base.base method), 24
ascii() (in module seisflows.plugins.readers), 22	check() (seisflows.preprocess.double_difference.double_difference
ascii() (in module seisflows.plugins.writers), 23	method), 25
	check () (seisflows.solver.base.base method), 26

check() (seisflows.solver.specfem2d.specfem	2 <i>d</i> 29
method), 29	<pre>check_source_names() (seisflows.solver.base.base</pre>
check() (seisflows.solver.specfem3d.specfem	
method), 29	check_status() (seis-
check () (seisflows.system.base.base method), 30	flows.plugins.optimize.LBFGS.LBFGS
check () (seisflows.system.lsf_lg.lsf_lg method), 31	method), 18
check() (seisflows.system.multicore.multico	
method), 32	flows.plugins.optimize.LCG.LCG method),
check () (seisflows.system.multithreaded.multithread	
method), 32	check_status() (seis-
check () (seisflows.system.pbs_lg.pbs_lg method), 33	
check () (seisflows.system.pbs_sm.pbs_lg method), 3	
check () (seisflows system serial serial method), 34	checkpoint() (seisflows.system.base.base method),  1). 30
<pre>check() (seisflows.system.slurm_lg.slurm_lg method 35</pre>	
	checkpoint() (seisflows.workflow.base.base method), 41
check() (seisflows.system.slurm_sm.slurm_, method), 36	checkpoint() (seisflows.workflow.inversion.inversion
check() (seisflows.system.tiger_lg.tiger_lg method),	
check() (seisflows.system.tiger_ig.tiger_ig method), check() (seisflows.system.tiger_sm.tiger_sm method	
36	clean() (seisflows.workflow.inversion.inversion
check () (seisflows.workflow.base.base method), 41	method), 42
check () (seisflows.workflow.inversion.inversi	
method), 42	method), 45
check () (seisflows.workflow.migration.migrati	
method), 43	flows.plugins.line_search.base.Base method),
check() (seisflows.workflow.test_adjoint.test_adjo	
method), 43	combine() (seisflows.solver.base.base method), 27
check() (seisflows.workflow.test_forward.test_forward	
method), 44	flows.optimize.base.base method), 15
check() (seisflows.workflow.test_postprocess.test_po	
method), 44	flows.optimize.LBFGS.LBFGS method),
check() (seisflows.workflow.test_preprocess.test_pre	process 16
method), 44	compute_direction() (seis-
check() (seisflows.workflow.test_system.test_syste	em flows.optimize.NLCG.NLCG method), 16
method), 45	compute_direction() (seis-
check_conjugacy() (in module se	is- flows.optimize.steepest_descent.steepest_descent
flows.plugins.optimize.NLCG), 19	method), 17
	is- compute_direction() (seis-
flows.plugins.optimize.NLCG), 19	flows.workflow.inversion.inversion method),
<pre>check_filter() (seisflows.preprocess.base.ba</pre>	
method), 24	config() (in module seisflows.config), 45
check_mesh_properties() (se	` '
flows.solver.base.base method), 26	copy_slice() (in module seis-
check_mute() (seisflows.preprocess.base.ba	* * * * * * * * * * * * * * * * * * * *
method), 24	correlate() (in module seisflows.tools.signal), 39
check_normalize() (seisflows.preprocess.base.ba	
method), 24	cp() (in module seisflows.tools.unix), 41
check_solver_parameter_files() (se	<u> </u>
flows.solver.base.base method), 27	24
check_solver_parameter_files() (se	
flows.solver.specfem2d.specfem2d method 29	d), cwd (seisflows.solver.base.base attribute), 27
check_solver_parameter_files() (se	-: <sub>5</sub> - D
flows.solver_parameter_files() (se	–
jiows.soiven.specjenisu.specjenisu memol	d), data filenames (seisflows.solver.base.base at-

tribute), 27	<pre>export_model() (seisflows.solver.base.base method),</pre>
data_filenames (seis-	27
flows.solver.specfem2d.specfem2d attribute), 29	<pre>export_model()</pre>
data_filenames (seis-	29
flows.solver.specfem3d.specfem3d attribute), 29	<pre>export_residuals() (seisflows.solver.base.base method), 27</pre>
data_wildcard (seis-	<pre>export_traces() (seisflows.solver.base.base</pre>
flows.solver.specfem3d.specfem3d attribute), 30	method), 27
default (class in seisflows.postprocess.default), 24	F
default (class in seisflows.preprocess.default), 25	finalize() (seisflows.workflow.inversion.inversion
Diagonal (class in seis-	method), 42
flows.plugins.preconds.diagonal), 20	finalize_search() (seisflows.optimize.base.base
Dict (class in seisflows.config), 45	method), 15
diff() (in module seisflows.tools.tools), 40	findpath() (in module seisflows.tools.tools), 40
Displacement() (in module seis-	fletcher_reeves() (in module seis-
flows.plugins.adjoint), 21	flows.plugins.optimize.NLCG), 19
Displacement () (in module seisflows.plugins.misfit),	forward() (seisflows.solver.base.base method), 27
21	forward() (seisflows.solver.specfem2d.specfem2d
distance() (seisflows.preprocess.double_difference.dou	
method), 25	forward() (seisflows.solver.specfem3d.specfem3d
divides () (in module seisflows.tools.tools), 40	method), 30
dot () (in module seisflows.tools.math), 38	
dot () (seisflows.optimize.base.base method), 15	G
DotProductLHS() (in module seis-	gabor() (in module seisflows.plugins.wavelets), 22
flows.workflow.test_adjoint), 43	gauss2 () (in module seisflows.tools.math), 38
DotProductRHS() (in module seis-	generate_data() (seisflows.solver.base.base
flows.workflow.test_adjoint), 43	method), 27
double_difference (class in seis-	generate_data() (seis-
$flows.preprocess.double\_difference), 25$	flows.solver.specfem2d.specfem2d method),
_	29
E	generate_data() (seis-
Envelope() (in module seisflows.plugins.adjoint), 21	flows.solver.specfem3d.specfem3d method),
Envelope() (in module seisflows.plugins.misfit), 21	30
Envelope2() (in module seisflows.plugins.adjoint), 21	<pre>generate_mesh() (seisflows.solver.base.base</pre>
Envelope2() (in module seisflows.plugins.misfit), 22	method), 27
Envelope3() (in module seisflows.plugins.adjoint), 21	generate_mesh() (seis-
Envelope3() (in module seisflows.plugins.misfit), 22	flows.solver.specfem2d.specfem2d method),
eval_func() (seisflows.solver.base.base method), 27	29
eval_func() (seisflows.solver.specfem3d.specfem3d	generate_mesh() (seis-
method), 30	flows.solver.specfem3d.specfem3d method),
eval_grad() (seisflows.solver.base.base method), 27	30
evaluate_function() (seis-	<pre>get_network_size() (seis-</pre>
flows.workflow.inversion.inversion method),	flows.preprocess.base.base method), 24
42	<pre>get_receiver_coords() (seis-</pre>
evaluate_gradient() (seis-	flows.preprocess.base.base method), 24
flows.workflow.inversion.inversion method),	<pre>get_regular_ticks() (in module seis-</pre>
42	flows.tools.graphics), 37
event (seisflows.workflow.test_adjoint.test_adjoint at-	<pre>get_source_coords() (seis-</pre>
tribute), 43	flows.preprocess.base.base method), 24
exists() (in module seisflows.tools.tools), 40	<pre>get_time_scheme() (seisflows.preprocess.base.base</pre>
export_kernels() (seisflows.solver.base.base	method), 24
method), 27	getpar() (in module seisflows.tools.seismic), 39

getset() (in module seisflows.tools.tools), 40 grad() (in module seisflows.tools.math), 38	io (seisflows.solver.base.base attribute), 28 iterable() (in module seisflows.tools.tools), 40
grid2mesh() (in module seisflows.tools.array), 37 gridsmooth() (in module seisflows.tools.array), 37	J
Н	<pre>job_array_args() (seisflows.system.pbs_lg.pbs_lg</pre>
hello() (seisflows.workflow.test_system.test_system method), 45	<pre>job_array_cmd() (seisflows.system.pbs_lg.pbs_lg</pre>
hilbert() (in module seisflows.tools.math), 38	job_array_status() (seis-
hostlist() (seisflows.system.pbs_sm.pbs_lg method),	flows.system.pbs_lg.pbs_lg method), 33
33 hostlist() (seisflows.system.slurm_sm.slurm_sm	job_array_status() (seis- flows.system.slurm_lg.slurm_lg method),
method), 36	35
hostname() (in module seisflows.tools.unix), 41	<pre>job_id_list()</pre>
	job_id_list() (seisflows.system.slurm_lg.slurm_lg
<pre>import_model() (seisflows.solver.base.base method),</pre>	method), 35
27	<pre>job_status() (seisflows.system.lsf_lg.lsf_lg method),</pre>
import_model() (seis-	31
flows.solver.specfem2d.specfem2d method), 29	<pre>job_status() (seisflows.system.slurm_lg.slurm_lg</pre>
<pre>import_traces()</pre>	K
initialize() (seis-	kernel_databases (seisflows.solver.base.base at-
flows.plugins.line_search.base.Base method),	tribute), 28
18	kernel_databases (seis-
initialize() (seisflows.plugins.optimize.LCG.LCG method), 19	flows.solver.specfem2d.specfem2d attribute), 29
initialize() (seisflows.workflow.inversion.inversion	kernel_databases (seis-
method), 42 initialize() (seis-	flows.solver.specfem3d.specfem3d attribute), 30
flows.workflow.thrifty_inversion.thrifty_inversion	
method), 45	L
initialize_adjoint_traces() (seis-	LBFGS (class in seisflows.optimize.LBFGS), 16
flows.solver.base.base method), 27	LBFGS (class in seisflows.plugins.optimize.LBFGS), 18
initialize_adjoint_traces() (seis-	LBFGS_(class in seisflows.plugins.optimize.PLCG), 19
flows.solver.specfem2d.specfem2d method),	LCG (class in seisflows.plugins.optimize.LCG), 19
29	line_search() (seis-
<pre>initialize_adjoint_traces() (seis- flows.solver.specfem3d.specfem3d method),</pre>	flows.workflow.inversion.inversion method), 42
30	ln () (in module seisflows.tools.unix), 41
initialize_search() (seis-	load () (in module seisflows.config), 46
flows.optimize.base.base method), 15	load() (seisflows.optimize.base.base method), 15
initialize_solver_directories() (seis-	load() (seisflows.solver.base.base method), 28
flows.solver.base.base method), 27	load_weights() (seis-
InstantaneousPhase() (in module seisflows.plugins.adjoint), 21	flows.preprocess.double_difference.double_difference method), 25
InstantaneousPhase() (in module seis-	loadjson() (in module seisflows.tools.tools), 40
flows.plugins.misfit), 22	loadnpy () (in module seisflows.tools.array), 37
InstantaneousPhase2() (in module seis-	loadnpy() (in module seisflows.tools.tools), 40
flows.plugins.adjoint), 21	loadobj () (in module seisflows.tools.tools), 40
InstantaneousPhase2() (in module seis-	loadpy () (in module seisflows tools tools), 40
flows.plugins.misfit), 22 inversion (class in seisflows.workflow.inversion), 42	loadtxt() (in module seisflows.tools.tools), 40 loadtxt() (seisflows.optimize.base.base method), 15
LII V CL CLOII (Class in scisjiows. Workjiow.inversion), 72	- Caacia () (being to in a copium Le touse touse me mou), 13

loadyaml () (in module seisflows.tools.tools), 40	multicore (class in seisflows.system.multicore), 32
ls () (in module seisflows.tools.unix), 41	multithreaded (class in seis-
lsf_lg (class in seisflows.system.lsf_lg), 31	flows.system.multithreaded), 32
lsq2() (in module seisflows.tools.math), 38	<pre>mute_early_arrivals() (in module seis- flows.tools.signal), 39</pre>
M	mute_late_arrivals() (in module seis-
main() (seisflows.workflow.base.base method), 41	flows.tools.signal), 39
main() (seisflows.workflow.inversion.inversion method), 42	<pre>mute_long_offsets() (in module seis- flows.tools.signal), 40</pre>
main() (seisflows.workflow.migration.migration method), 43	<pre>mute_short_offsets() (in module seis- flows.tools.signal), 40</pre>
main() (seisflows.workflow.test_adjoint.test_adjoint method), 43	mv () (in module seisflows.tools.unix), 41
main() (seisflows.workflow.test_forward.test_forward	N
method), 44	nabla() (in module seisflows.tools.math), 38
main() (seisflows.workflow.test_postprocess.test_postpro	
method), 44	newline() (seisflows.plugins.line_search.base.Writer
main() (seisflows.workflow.test_preprocess.test_preproce	
method), 44	NLCG (class in seisflows.optimize.NLCG), 16
main() (seisflows.workflow.test_system.test_system	NLCG (class in seisflows.plugins.optimize.NLCG), 19
method), 45	nproc() (in module seisflows.tools.tools), 40
mask () (in module seisflows.tools.signal), 39	Null (class in seisflows.config), 45
merge() (seisflows.solver.base.base method), 28 mesh2grid() (in module seisflows.tools.array), 37	P
mesh_properties (seisflows.solver.base.base at-	na alreade avi et e () (in module seisflaus toels toels)
tribute), 28	package_exists() (in module seisflows.tools.tools), 40
meshsmooth() (in module seisflows.tools.array), 37	ParameterError, 37
migration (class in seisflows.workflow.migration), 43	pbs_lg (class in seisflows.system.pbs_lg), 32
Minmax (class in seisflows.tools.seismic), 39	pbs_lg (class in seisflows.system.pbs_sm), 33
mkdir() (in module seisflows.tools.unix), 41	pkgpath() (in module seisflows.tools.tools), 40
model_cutoff() (seisflows.optimize.base.base	PLCG (class in seisflows.plugins.optimize.PLCG), 19
method), 15	<pre>plot_gll() (in module seisflows.tools.graphics), 37</pre>
model_databases (seisflows.solver.base.base at-	plot_many_gll() (in module seis-
tribute), 28	flows.tools.graphics), 37
model_databases (seis-	plot_section() (in module seis-
flows.solver.specfem2d.specfem2d attribute),	flows.tools.graphics), 37
29	<pre>plot_vector() (in module seisflows.tools.graphics),</pre>
model_databases (seis-	38
flows.solver.specfem3d.specfem3d attribute), 30	pollak_ribere() (in module seis-
module_exists() (in module seisflows.tools.tools),	flows.plugins.optimize.NLCG), 19
40	polyfit2() (in module seisflows.tools.math), 38
mpiexec() (seisflows.system.lsf_lg.lsf_lg method), 31	prepare_eval_grad() (seis-
mpiexec() (seisflows.system.pbs_lg.pbs_lg method),	flows.preprocess.base.base method), 24 prepare_model() (seis-
33	prepare_model() (seis- flows.workflow.migration.migration method),
<pre>mpiexec() (seisflows.system.pbs_sm.pbs_lg method),</pre>	43
33	prepare_model() (seis-
mpiexec() (seisflows.system.serial.serial method), 34	flows.workflow.test_adjoint.test_adjoint
mpiexec() (seisflows.system.slurm_lg.slurm_lg	method), 43
method), 35	process_kernels() (seis-
<pre>mpiexec() (seisflows.system.slurm_sm.slurm_sm</pre>	flows.postprocess.base.base method), 23
method), 36	progress() (seisflows.system.serial.serial method),
mread() (in module seisflows.plugins.solver_io.adios),	34
20	

R	S
read() (in module seisflows.plugins.solver_io.adios),	save() (in module seisflows.config), 46
20	save() (seisflows.optimize.base.base method), 16
read_slice() (in module seis-	save() (seisflows.solver.base.base method), 28
$flows.plugins.solver\_io.fortran\_binary), 21$	<pre>save() (seisflows.workflow.test_preprocess.test_preprocess</pre>
readBigSuFile() (in module seis-	method), 44
flows.plugins.readers), 22	save_gradient() (seis-
cename () (in module seisflows.tools.unix), 41	flows.workflow.inversion.inversion method),
cename_data() (seisflows.solver.base.base method),	42
28	save_kernels() (seis-
rename_data() (seis-	flows.workflow.inversion.inversion method),
flows.solver.specfem3d.specfem3d method), 30	42 save kernels() (seis-
rename_kernels() (seisflows.solver.base.base	save_kernels() (seis- flows.workflow.migration.migration method),
method), 28	43
restart() (seisflows.optimize.base.base method), 15	<pre>save_kernels_sum() (seis-</pre>
restart() (seisflows.optimize.LBFGS.LBFGS method), 16	flows.workflow.migration.migration method), 43
restart() (seisflows.optimize.NLCG.NLCG method),	save_kwargs() (seisflows.system.lsf_lg.lsf_lg
16	method), 31
restart () (seisflows.optimize.steepest_descent.steepest method), 17	_desventkwargs() (seisflows.system.multicore.multicore method), 32
restart() (seisflows.plugins.optimize.LBFGS.LBFGS	save_kwargs() (seisflows.system.pbs_lg.pbs_lg
method), 18	method), 33
restart() (seisflows.plugins.optimize.NLCG.NLCG method), 19	save_kwargs() (seisflows.system.pbs_sm.pbs_lg method), 34
restarted(seisflows.optimize.steepest_descent.steepest	
attribute), 17	flows.system.slurm_sm.slurm_sm method),
retry_status() (seisflows.optimize.base.base	36
method), 15	<pre>save_model() (seisflows.workflow.inversion.inversion</pre>
ricker() (in module seisflows.plugins.wavelets), 22	method), 42
cm () (in module seisflows.tools.unix), 41	save_residuals() (seis-
run () (seisflows.system.base.base method), 30	flows.workflow.inversion.inversion method),
run () (seisflows.system.lsf_lg.lsf_lg method), 31	42
run () (seisflows.system.multicore.multicore method), 32	save_traces() (seis-
run () (seisflows.system.pbs_lg.pbs_lg method), 33	flows.workflow.inversion.inversion method), 42
run () (seisflows.system.pbs_tg.pbs_lg method), 34	save_traces() (seis-
run () (seisflows.system.serial.serial method), 34	flows.workflow.migration.migration method),
run () (seisflows.system.slurm_lg.slurm_lg method), 35	43
run () (seisflows.system.slurm_sm.slurm_sm method),	savejson() (in module seisflows.tools.tools), 41
36	savenpy() (in module seisflows.tools.array), 37
run_single() (seisflows.system.base.base method),	savenpy() (in module seisflows.tools.tools), 41
30	saveobj() (in module seisflows.tools.tools), 41
run_single() (seisflows.system.lsf_lg.lsf_lg method),	savetxt() (in module seisflows.tools.tools), 41
31	<pre>savetxt() (seisflows.optimize.base.base method), 16</pre>
run_single() (seisflows.system.multicore.multicore	sconvolve() (in module seisflows.tools.signal), 40
method), 32	search_history() (seis-
run_single() (seisflows.system.serial.serial method), 34	flows.plugins.line_search.base.Base method), 18
run_single() (seisflows.system.slurm_lg.slurm_lg	seisflows.config(module),45
method), 35	seisflows.optimize.base (module), 15
run_single() (seisflows.system.slurm_sm.slurm_sm	seisflows.optimize.LBFGS (module), 16
method), 36	seisflows.optimize.NLCG(module), 16

```
seisflows.optimize.steepest_descent
                                               seisflows.system.tiger_sm(module), 36
        (module), 17
                                               seisflows.tools.array (module), 37
seisflows.plugins.adjoint (module), 21
                                               seisflows.tools.err(module), 37
seisflows.plugins.line_search.backtrack seisflows.tools.graphics(module), 37
        (module), 17
                                               seisflows.tools.math (module), 38
seisflows.plugins.line search.base(mod-
                                               seisflows.tools.msg(module), 39
                                               seisflows.tools.seismic (module), 39
                                               seisflows.tools.signal(module), 39
seisflows.plugins.line_search.bracket
        (module), 18
                                               seisflows.tools.tools(module), 40
seisflows.plugins.misfit (module), 21
                                               seisflows.tools.unix (module), 41
seisflows.plugins.optimize.LBFGS
                                         (mod-
                                               seisflows.workflow.base(module),41
                                               seisflows.workflow.inversion(module), 42
        ule), 18
seisflows.plugins.optimize.LCG
                                     (module),
                                               seisflows.workflow.migration(module), 43
                                               seisflows.workflow.test_adjoint (module),
        19
seisflows.plugins.optimize.NLCG (module),
                                               seisflows.workflow.test_forward (module),
seisflows.plugins.optimize.PLCG (module),
                                               seisflows.workflow.test_postprocess
seisflows.plugins.preconds.diagonal
                                                       (module), 44
        (module), 20
                                               seisflows.workflow.test preprocess (mod-
seisflows.plugins.readers (module), 22
                                                       ule), 44
seisflows.plugins.solver.specfem2d(mod-
                                               seisflows.workflow.test_system (module),
        ule), 20
                                                       45
seisflows.plugins.solver.specfem3d(mod-
                                               seisflows.workflow.thrifty inversion
       ule), 20
                                                       (module), 45
seisflows.plugins.solver.specfem3d_globeselect() (in module seisflows.tools.unix), 41
        (module), 20
                                               serial (class in seisflows.system.serial), 34
                                               setpar() (in module seisflows.tools.seismic), 39
seisflows.plugins.solver_io.adios (mod-
                                               setup() (seisflows.optimize.base.base method), 16
       ule), 20
seisflows.plugins.solver_io.fortran_binasetup() (seisflows.optimize.LBFGS.LBFGS method),
        (module), 21
seisflows.plugins.wavelets(module), 22
                                               setup() (seisflows.optimize.NLCG.NLCG method), 16
                                               setup() (seisflows.optimize.steepest_descent.steepest_descent
seisflows.plugins.writers(module), 23
seisflows.postprocess.base (module), 23
                                                       method), 17
seisflows.postprocess.default (module), 24
                                               setup() (seisflows.postprocess.base.base method), 23
seisflows.preprocess.base (module), 24
                                               setup() (seisflows.preprocess.base.base method), 24
seisflows.preprocess.default (module), 25
                                               setup() (seisflows.solver.base.base method), 28
seisflows.preprocess.double_difference
                                               setup()
                                                              (seis flows.work flow.inversion.inversion
        (module), 25
                                                       method), 43
seisflows.solver.base (module), 26
                                               shift() (seisflows.preprocess.double_difference.double_difference
seisflows.solver.specfem2d(module), 29
                                                       method), 25
seisflows.solver.specfem3d (module), 29
                                               slurm lq (class in seisflows.system.slurm lg), 34
                                               slurm sm (class in seisflows.system.slurm sm), 35
seisflows.system.base (module), 30
                                               smooth () (seisflows.solver.base.base method), 28
seisflows.system.lsf_lg(module), 31
seisflows.system.lsf_sm(module), 32
                                               smooth_legacy()
                                                                      (in
                                                                                         seis-
seisflows.system.multicore (module), 32
                                                       flows.plugins.solver.specfem2d), 20
seisflows.system.multithreaded (module),
                                               sortrows () (in module seisflows.tools.array), 37
                                               source_name (seisflows.solver.base.base attribute), 28
seisflows.system.pbs_lg(module), 32
                                               source_names (seisflows.solver.base.base attribute),
seisflows.system.pbs_sm(module), 33
seisflows.system.serial(module), 34
                                               source_prefix (seisflows.solver.base.base attribute),
seisflows.system.slurm_lg(module), 34
                                                       28
seisflows.system.slurm_sm (module), 35
                                               source_prefix
                                                                                         (seis-
seisflows.system.tiger_lg(module), 36
                                                       flows.solver.specfem2d.specfem2d
                                                                                     attribute),
```

29	test_filter() (seis-
source_prefix (seis-	flows.workflow.test_preprocess.test_preprocess
flows.solver.specfem3d.specfem3d attribute),	method), 44
30	test_forward (class in seis-
specfem2d (class in seisflows.solver.specfem2d), 29	flows.workflow.test_forward), 44
specfem3d (class in seisflows.solver.specfem3d), 29	test_misfit() (seis-
split () (seisflows.solver.base.base method), 28	flows.workflow.test_preprocess.test_preprocess
stack() (in module seisflows.tools.array), 37	method), 44
status (seisflows.workflow.thrifty_inversion.thr	stomst_mute() (seisflows.workflow.test_preprocess.test_preprocess method), 44
	test_normalize() (seis-
flows.optimize.steepest_descent), 17	flows.workflow.test_preprocess.test_preprocess
Struct (class in seisflows.tools.tools), 40	method), 44
su() (in module seisflows.plugins.readers), 22	test_postprocess (class in seis-
su() (in module seisflows.plugins.writers), 23	flows.workflow.test_postprocess), 44
submit () (seisflows.system.base.base method), 31	test_preprocess (class in seis-
<pre>submit() (seisflows.system.lsf_lg.lsf_lg method), 31</pre>	flows.workflow.test_preprocess), 44
<pre>submit() (seisflows.system.multicore.multicore</pre>	test_reader() (seis-
method), 32	flows.workflow.test_preprocess.test_preprocess
<pre>submit() (seisflows.system.pbs_lg.pbs_lg method), 33</pre>	method), 44
<pre>submit() (seisflows.system.pbs_sm.pbs_lg method), 34</pre>	test_system (class in seis-
submit() (seisflows.system.serial.serial method), 34	flows.workflow.test_system), 45
<pre>submit() (seisflows.system.slurm_lg.slurm_lg</pre>	test_writer() (seis-
method), 35	flows.workflow.test_preprocess.test_preprocess
<pre>submit() (seisflows.system.slurm_sm.slurm_sm</pre>	method), 44
method), 36	thrifty_inversion (class in seis-
<pre>submit() (seisflows.system.tiger_lg.tiger_lg method),</pre>	flows.workflow.thrifty_inversion), 45
36	tiger_lg (class in seisflows.system.tiger_lg), 36
submit() (seisflows.system.tiger_sm.tiger_sm	tiger_sm(class in seisflows.system.tiger_sm), 36
method), 36	tilde_expand() (in module seisflows.config), 46
submit_job_array() (seis-	timestamp() (in module seisflows.tools.tools), 41
flows.system.pbs_lg.pbs_lg method), 33	timestamp() (seisflows.system.lsf_lg.lsf_lg method),
sum_residuals() (seisflows.preprocess.base.base	31
method), 25	touch() (in module seisflows.tools.unix), 41
sum_residuals() (seis-	Traveltime() (in module seisflows.plugins.adjoint),
flows.preprocess.double_difference.double_differ	
method), 25	Traveltime() (in module seisflows.plugins.misfit), 22
Т	TraveltimeInexact() (in module seis-flows.plugins.adjoint), 21
taskid (seisflows.solver.base.base attribute), 28	TraveltimeInexact() (in module seis-
taskid() (seisflows.system.base.base method), 31	flows.plugins.misfit), 22
<pre>taskid() (seisflows.system.lsf_lg.lsf_lg method), 31</pre>	tukeywin() (in module seisflows.tools.signal), 40
<pre>taskid() (seisflows.system.pbs_lg.pbs_lg method), 33</pre>	tv() (in module seisflows.tools.math), 38
taskid() (seisflows.system.pbs_sm.pbs_lg method), 34	11
taskid() (seisflows.system.serial.serial method), 34	U
taskid() (seisflows.system.slurm_lg.slurm_lg	uniquerows () (in module seisflows.tools.array), 37
method), 35	update() (seisflows.config.Dict method), 45
taskid() (seisflows.system.slurm_sm.slurm_sm method), 36	update() (seisflows.plugins.line_search.base.Base method), 18
test_adjoint (class in seis-	update() (seisflows.plugins.optimize.LBFGS.LBFGS
flows.workflow.test_adjoint), 43	method), 18
test_adjoint() (seis-	update() (seisflows.plugins.optimize.LCG.LCG
flows.workflow.test_preprocess.test_preprocess	method), 19
method), 44	update() (seisflows.tools.seismic.Minmax method), 39

```
update_search()
                         (seisflows.optimize.base.base write slice()
                                                                               (in
                                                                                         module
                                                                                                      seis-
        method), 16
                                                               flows.plugins.solver_io.fortran_binary), 21
update_status()
                                               (seis-
                                                      write sources()
                                                                                 (in
                                                                                          module
                                                                                                      seis-
        flows.workflow.thrifty_inversion.thrifty_inversion
                                                               flows.plugins.solver.specfem2d), 20
        method), 45
                                                      write sources()
                                                                                 (in
                                                                                          module
                                                                                                      seis-
                                                               flows.plugins.solver.specfem3d), 20
V
                                                                                          module
                                                      write sources()
                                                                                 (in
                                                                                                      seis-
                                                               flows.plugins.solver.specfem3d_globe), 20
Velocity () (in module seisflows.plugins.adjoint), 21
                                                      write sources()
                                                                                                      (seis-
Velocity() (in module seisflows.plugins.misfit), 22
                                                               flows.solver.specfem3d.specfem3d
                                                                                                   method),
W
                                                      writeBigSuFile()
                                                                                  (in
                                                                                          module
                                                                                                      seis-
Waveform() (in module seisflows.plugins.adjoint), 21
                                                               flows.plugins.writers), 23
Waveform() (in module seisflows.plugins.misfit), 22
                                                      Writer (class in seisflows.plugins.line_search.base), 18
which () (in module seisflows.tools.unix), 41
                                                      Writer (class in seisflows.tools.seismic), 39
write() (in module seisflows.plugins.solver io.adios),
write_adjoint_traces()
                                               (seis-
        flows.preprocess.base.base method), 25
write_adjoint_traces()
                                               (seis-
        flows.preprocess.double_difference.double_difference
        method), 26
write_gradient() (seisflows.postprocess.base.base
        method), 23
write_gradient()
                                               (seis-
        flows.workflow.inversion.inversion
                                            method),
        43
                                               (seis-
write header()
        flows.plugins.line_search.base.Writer method),
write_misfit()
                                               (seis-
        flows.workflow.inversion.inversion
                                            method),
write_model()
                                               (seis-
        flows.workflow.inversion.inversion
                                            method),
        43
write_parameters()
                             (in
                                                seis-
        flows.plugins.solver.specfem3d_globe), 20
write_parameters()
                                               (seis-
        flows.solver.specfem3d.specfem3d
                                            method),
         30
write_receivers()
                            (in
                                    module
                                                seis-
        flows.plugins.solver.specfem2d), 20
write receivers()
                                    module
                            (in
                                                seis-
        flows.plugins.solver.specfem3d), 20
write_receivers()
                                    module
                            (in
                                                seis-
        flows.plugins.solver.specfem3d_globe), 20
write_receivers()
                                               (seis-
        flows.solver.specfem3d.specfem3d
                                            method),
write_residuals() (seisflows.preprocess.base.base
         method), 25
write_residuals()
                                               (seis-
        flows.preprocess.double_difference.double_difference
```

method), 26