TIDES 2016 Practical - Part 2

Finite-frequency kernels for noise correlation measurements

During the morning session Laura presented how to get correlation functions from recordings of ambient noise and highlighted the most important processing steps as well as their effect. In addition, you got familiar with a source imaging technique that uses a ray approximation of finite-frequency source kernels.
In part 2 we first present a method to compute synthetic correlation functions that accounts for the distribution of noise sources, heterogeneous Earth structure and seismic wave propagation physics. We then drop the ray approximation used in part 1 and compute finite-frequency kernels, not only for the source distribution, but also for Earth structure.

For this purpose we use the package `fd2d_noise`[1]. It is based on adjoint techniques and on a 2D finite-difference discretization of the acoustic wave equation as an analogue for fundamental-mode surface wave propagation. The package is organized in the following way:

`code/`:	contains functions for wave-propagation and kernel calculation
`input/`:	collects the most important input functions
	study `input/input_parameters.m` carefully
`literature/`:	holds material for recapitulation and further reading
`output/`:	array, correlations and Green functions are saved here
`tools/`:	encloses functions for measurements, plotting and small helpers

For the instructions we use the following layout:

- scripts and folders are indicated in typewriter font, e.g. `folder/script.m`
- variables or code are highlighted with `green`
- optional parts are written in grey
- questions are formulated in *red*

IMPORTANT:

- first run `startup.m` in package folder `fd2d_noise`, whenever you start MATLAB
- whenever reference stations, velocity models or simulation parameters are changed, delete calculated Green functions in `output/`

[1] Please note that we use a small setup for the tutorial that allows us to perform the calculations rather quickly, with the downside that it tempts to exaggerate the effect of heterogeneous noise source distributions on travel-times.

Task 1 - Modelling noise correlation functions

The goal of this task is to model correlation functions for different noise source distributions and different velocity models to investigate the respective effects. Under the assumption of spatially uncorrelated noise sources, a correlation wavefield $C(\mathbf{x}, \mathbf{x}_i)$ for a reference station at $\mathbf{x}_i$ can be computed for arbitrary distributions of the noise power-spectral density $S(\mathbf{x}, \omega)$ and is sampled at any position where another receiver is located. In the frequency domain the correlation wavefield is given by

$$C(\mathbf{x}, \mathbf{x}_i, \omega) = \int_{\oplus} G(\mathbf{x}, \boldsymbol{\xi}, \omega) \left[G^*(\boldsymbol{\xi}, \mathbf{x}_i, \omega)\, S(\boldsymbol{\xi}, \omega) \right] d\boldsymbol{\xi}, \tag{1}$$

where $\oplus$ denotes the volume of the Earth and $G(\mathbf{x}, \mathbf{x}_s, \omega)$ a Green function for a source at $\mathbf{x}_s$. Equation (1) suggests the following recipe:

1. Model the Green function $G(\mathbf{x}, \mathbf{x}_i, \omega)$ with source at $\mathbf{x}_i$
2. Multiply its complex conjugate, equivalent to its time-reversed version in the time domain, with the power-spectral density $S(\mathbf{x}, \omega)$
3. Compute the correlation wavefield as a regular solution of the wave equation with $G^*(\mathbf{x}, \mathbf{x}_i, \omega)\, S(\mathbf{x}, \omega)$ as source

With `fd2d_noise`, we first compute correlation functions for a homogeneous distribution of the power-spectral density, which produces symmetric correlation functions. In the next two runs, we use a point-localized noise source and a homogeneous distribution with a Gaussian anomaly superimposed.

Look at the corresponding noise source distributions (see description of scripts below). What do you expect? Try to sketch the corresponding correlation functions before you compute them.

Steps:

1. `input/make_noise_source.m`:
   run script directly to visualize noise source distribution specified in `input/input_parameters.m` and frequency spectrum used to calculate correlation functions

2. `code/calculate_correlations.m`:

   - run script to calculate correlation functions for the noise source distribution and the velocity model chosen in `input/input_parameters.m`

     INFO: arrays and computed correlation functions are saved in `output` with a filename according to the following scheme:

     `array_nref-<total number of reference stations>.mat`

     `correlations_nref-<total number of reference stations>` ...
     ... `_model-<model_type>_source-<source_type>.mat`

   - identify the three steps given by the recipe above

   - understand definition of array and reference stations (use pre-defined array and reference station for first solution of tasks 1-3)

3. `input/input_parameters.m`:

   - use different options for `source_type`

   - use different velocity models, i.e. change `model_type`

4. `input/make_noise_source.m`: define your own noise source distribution

5. `input/define_material_parameters.m`:

   - design your own velocity model

   - run script directly to visualize velocity model

Task 2 - Kernels for noise source distribution

Now that we know how to calculate synthetic correlation functions given a source and an Earth model, we can compare them to observed correlation functions with the help of suitable misfit functionals in order to locate noise sources (task 2) and to infer knowledge about Earth structure (task 3).
For a minimization of observables based on gradient-based methods, such as steepest descent or BFGS, an efficient method to compute first derivatives is essential. In our case the number of model parameters is usually large and the solution of the forward problem is computationally relatively expensive. Simply perturbing each element of the model vector and evaluating the effect on the misfit is thus prohibitive. Therefore, we apply adjoint techniques (see e.g. Tromp et al. 2005) that allow us to compute the first variation of the misfit functional with respect to the model parameters efficiently.

The source kernel $K(\mathbf{x})$ for a reference station at $\mathbf{x}_i$ and a receiver at $\mathbf{x}_r$ is in the frequency domain given by

$$K(\mathbf{x}) = \int_{-\infty}^{\infty} f^*(\omega)\, G(\mathbf{x}, \mathbf{x}_r, \omega)\, G^*(\mathbf{x}, \mathbf{x}_i, \omega)\, d\omega, \tag{2}$$

with the adjoint source $f(\omega)$ (see e.g. Fichtner 2016). The term $f^*(\omega)\, G(\mathbf{x}, \mathbf{x}_r, \omega)$ can be interpreted as an adjoint wavefield, excited by $f(\omega)$ at $\mathbf{x}_r$, which interacts with the time-reversed Green function of the reference station.

As a first step to train our intuition, we perform data-independent measurements in `fd2d_noise`. That means we do not compare the synthetics to data yet, but we investigate where energy in specific time windows originates from. The signal is windowed, time reversed and re-injected at the receiver. The resulting kernel indicates, where noise sources can have an effect on the signal in the deployed time window.
In the second step, we use previously computed correlations as data and compute the first derivatives for two different misfit functionals, i.e. waveform differences and the measurement presented by Laura based on the logarithmic energy ratio of causal and anti-causal time windows.

Steps:

1. **data-independent mode:**

   - use the following settings

     - in `input/input_parameters.m`:
       `source_type = 'homogeneous'` and `model_type = 1`

     - in `code/calculate_kernels.m`

       ```
       usr_par.type = 'source';
       usr_par.measurement.type = 'waveform_difference';
       usr_par.measurement.mode = 'manual';
       usr_par.data_independent = 'yes';
       ```
       `array_file` and `data_file` are not of importance

   - run `code/calculate_kernels.m`

   - pick different time windows, e.g. from 10 to 50 seconds

     How does the kernel look like for a window from -5 to 5 seconds? Does it agree with the `source_type = 'point'`*? What would be different in 3D?*

2. **with synthetic 'data':**

   - use the following settings:

     - in `code/calculate_kernels.m`

       specify `array_file` and `data_file` computed in Task 1, e.g.
       ```
       array_file = 'array_nref-1.mat';
       data_file = 'correlations_nref-1_model-1_source-gaussian.mat';

       usr_par.type = 'source';
       usr_par.measurement.type = 'waveform_difference';
       usr_par.measurement.mode = 'manual' or 'auto';
       usr_par.data_independent = 'no';z
       ```

   - run `code/calculate_kernels.m`

     Does the gradient make sense? What has to be changed when using it as an update to the source model?

- run `code/calculate_kernels.m` again, but now with `'log_amplitude_ratio'` as misfit function (see Ermert et al. (2016))

  INFO: for the `log_amplitude_ratio` measurement only pick window on either the causal or acausal branch; the second window is internally mirrored onto the other branch

  What is different in comparison to the kernel with `'waveform_difference'` *as measurement? Can you explain the different nature of the kernel?*

- use different synthetic data in `code/calculate_kernels.m` and vary the initial source and/or velocity model in `input/input_parameters.m`

Task 3 - Kernels for Earth structure

For the inversion for Earth structure we are interested in the first variation of the misfit functional χ with respect to Earth model parameters, i.e. in

$$\delta\chi = \int_{\oplus} \int_{-\infty}^{\infty} u^{\dagger}_{(2)}(\mathbf{x},\omega)\, \delta\mathcal{L}(C(\mathbf{x},\mathbf{x}_2,\omega)) + u^{\dagger}_{(1)}(\mathbf{x},\omega)\, \delta\mathcal{L}(C(\mathbf{x},\mathbf{x}_1,\omega))\, d\omega\, d\mathbf{x}, \qquad (3)$$

where $u^{\dagger}_{(1)}$ and $u^{\dagger}_{(2)}$ denote the adjoint wavefields for the correlation functions at reference stations $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, and $\delta\mathcal{L}$ is the variation of the forward modeling operator with respect to Earth structure (see e.g. Fichtner 2014, 2016 for more details). The structure of Equation (3) is similar to the structure of $\delta\chi$ when a traditional source-receiver configuration is used: the variation is given by a product of a forward and an adjoint field, integrated over space and frequency - or over time, when working in the time domain. However, Equation (3) has two terms instead of only one as in traditional source-receiver configurations.

Steps:

1. **calculate data**
   - use the following settings in `input/input_parameters.m`:
     `model_type = 2` and `source_type = 'homogeneous'`
   - run `code/calculate_correlations.m`
2. **calculate kernel**
   - use the following settings:
     - in `code/calculate_kernels.m`:
       use array and data calculated in previous step, e.g.

```
array_file = 'array_nref-1.mat';
data_file = 'correlations_nref-1_model-2_source-homogeneous.mat';

usr_par.type = 'structure';
usr_par.measurement.type = 'waveform_difference';
usr_par.measurement.mode = 'manual' or 'auto';
usr_par.data_independent = 'no';
```

     - in `input/input_parameters.m`:
       `model_type = 1` and `source_type = 'homogeneous'`
   - run `code/calculate_kernels.m`
3. repeat step 1 and 2 for `source_type = 'gaussian'` and `source_type = 'point'` (do not forget to change data file in step 2 accordingly)

Task 4 - Design of an array for source inversions (optional)

In the previous tasks you calculated noise correlation functions for different noise source distributions as well as kernels for their distribution and for Earth structure. For training and computational purposes, however, we only used two stations and it is hard to imagine, how a noise source anomaly can be localized with the presented kernels.

In this task you calculate the first update for an iterative inversion procedure for a realistic array configuration and may optimize the array for a better location of the source anomaly.

Steps:

1. define a source distribution in `input/make_noise_source.m` that you would like to invert for, e.g. an anomaly in the center of the model domain.

2. design an array in `code/calculate_correlations.m` suitable to locate the anomaly; you can also use more reference stations to improve the coverage

   INFO:

   - to facilitate the design of the array, temporarily set a return-statement in `code/calculate_correlations.m` before the loop over the reference stations

   - with a multi-core processor consider using parfor instead of for in the loop over reference stations and set `make_plots = 'no'`)

3. compute the corresponding correlation functions

4. calculate the source kernel with different misfit functionals and with a homogeneous noise source distribution as initial model

5. try to improve the array configuration and the selection of the reference stations for a better localization of the source anomaly

What is the fundamental difference in the design of an array targeted for
a) an inversion for structure and
b) for the distribution of noise sources?