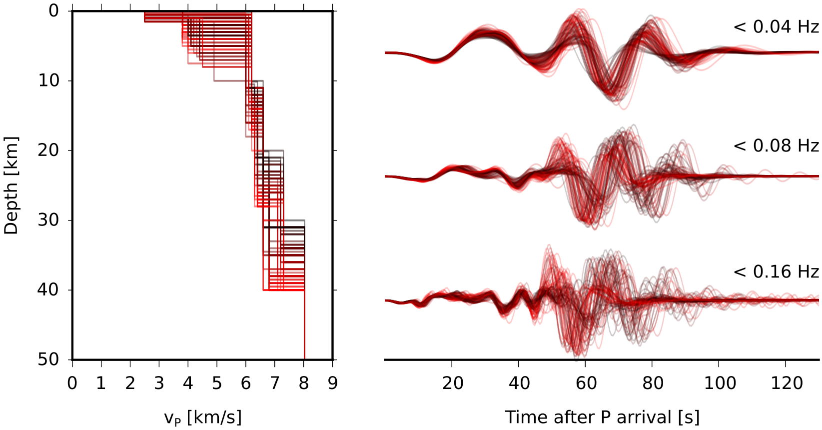
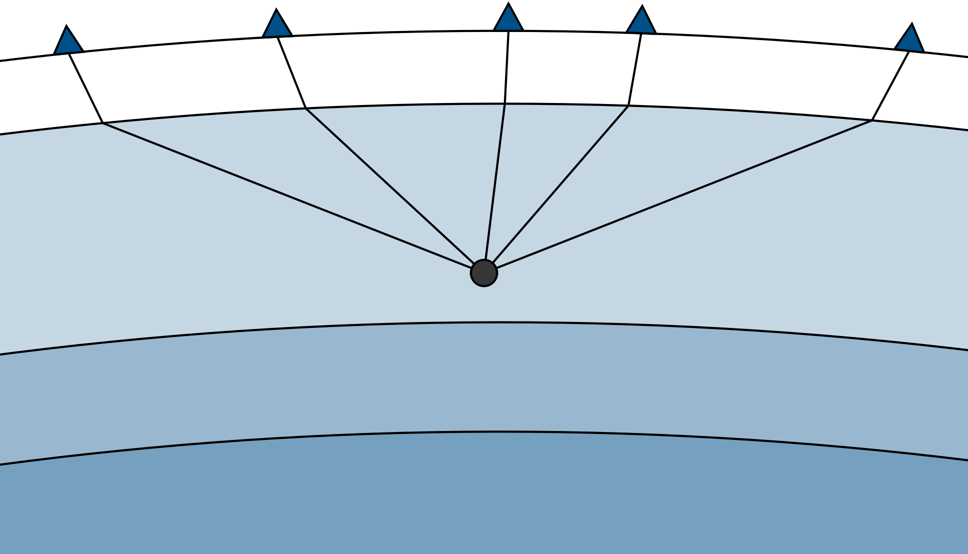
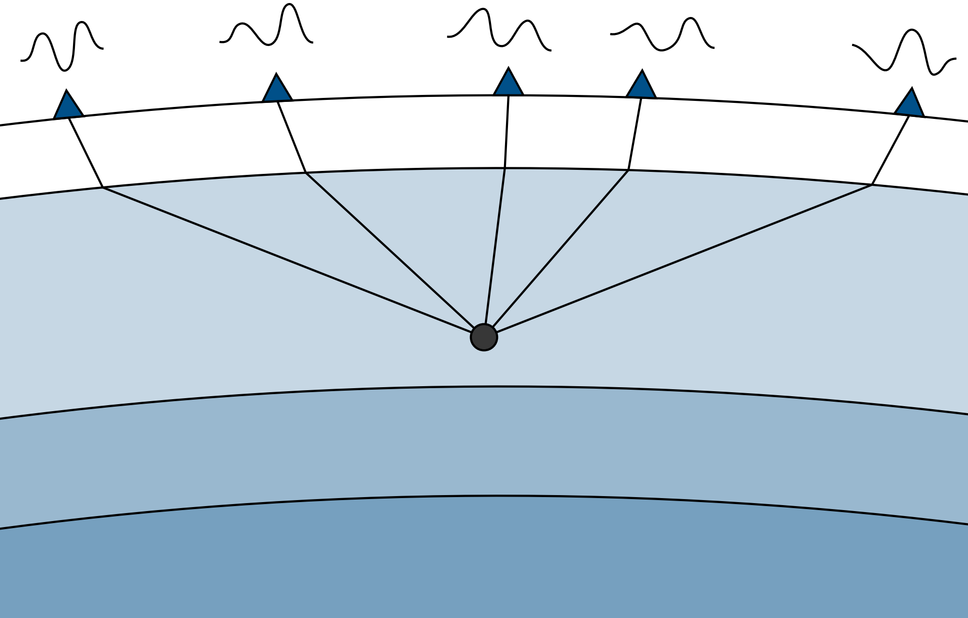
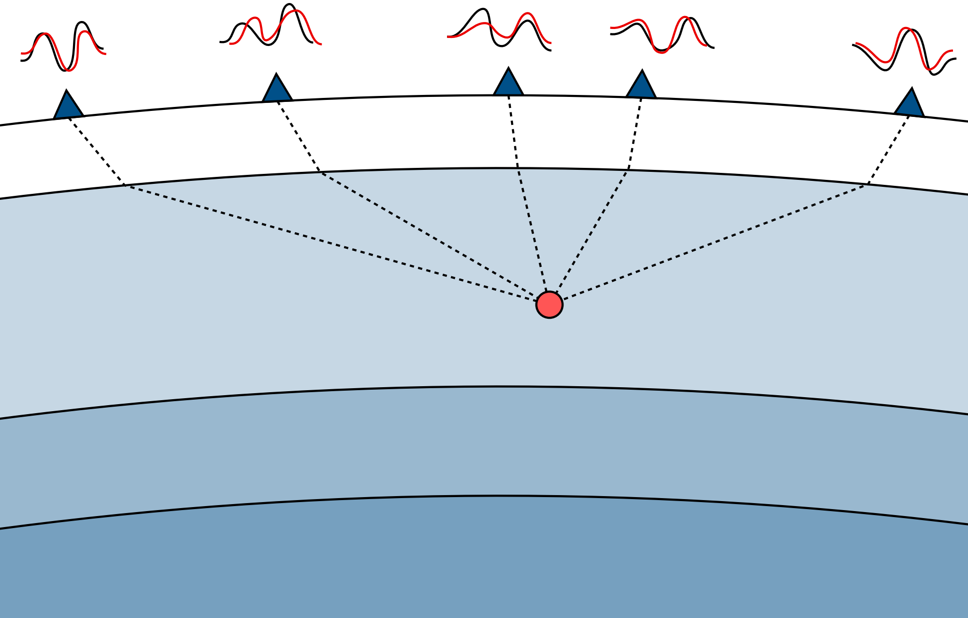
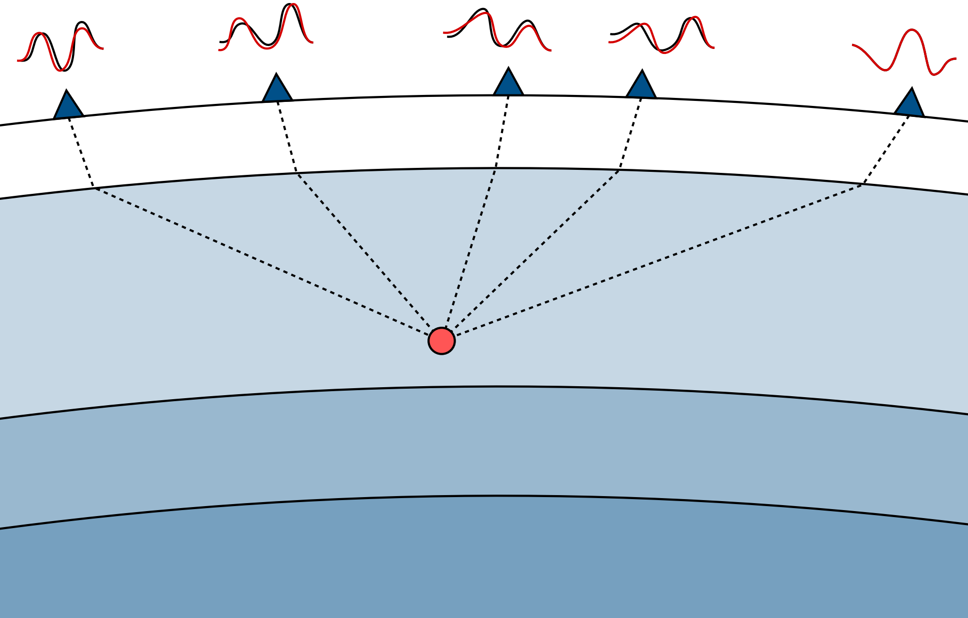
Green’s functions and synthetic seismogram generation

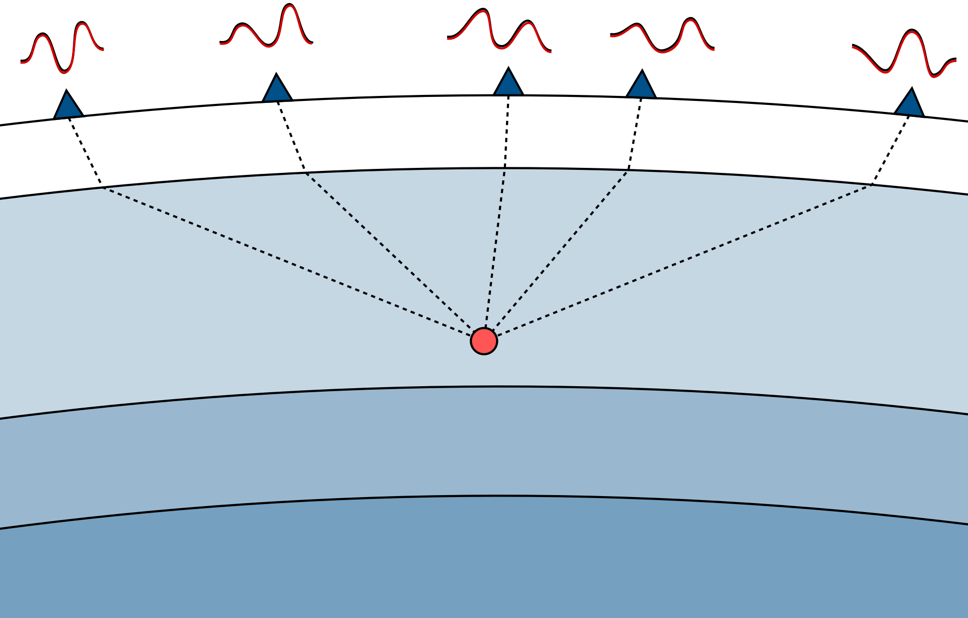


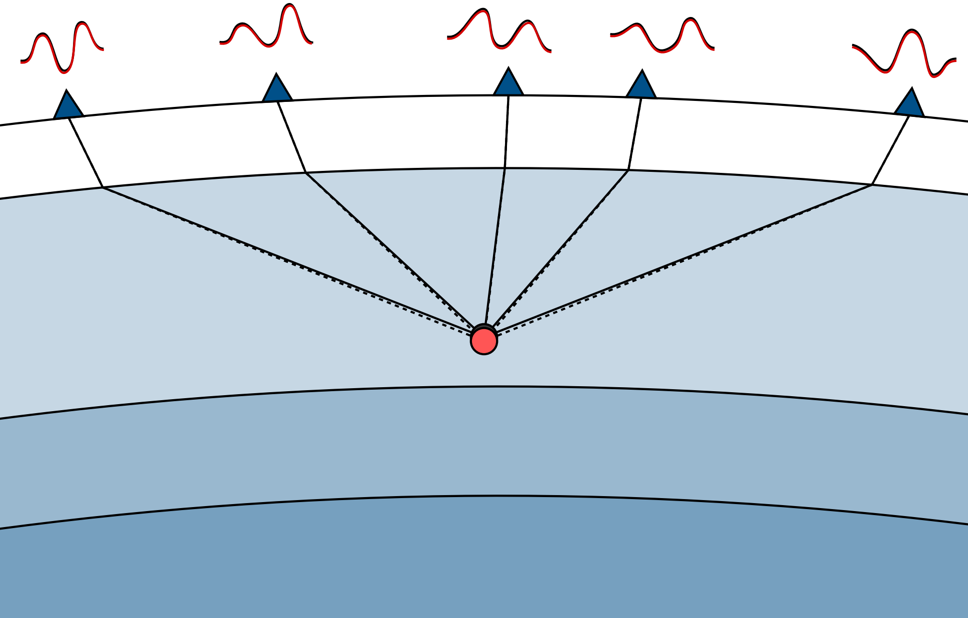


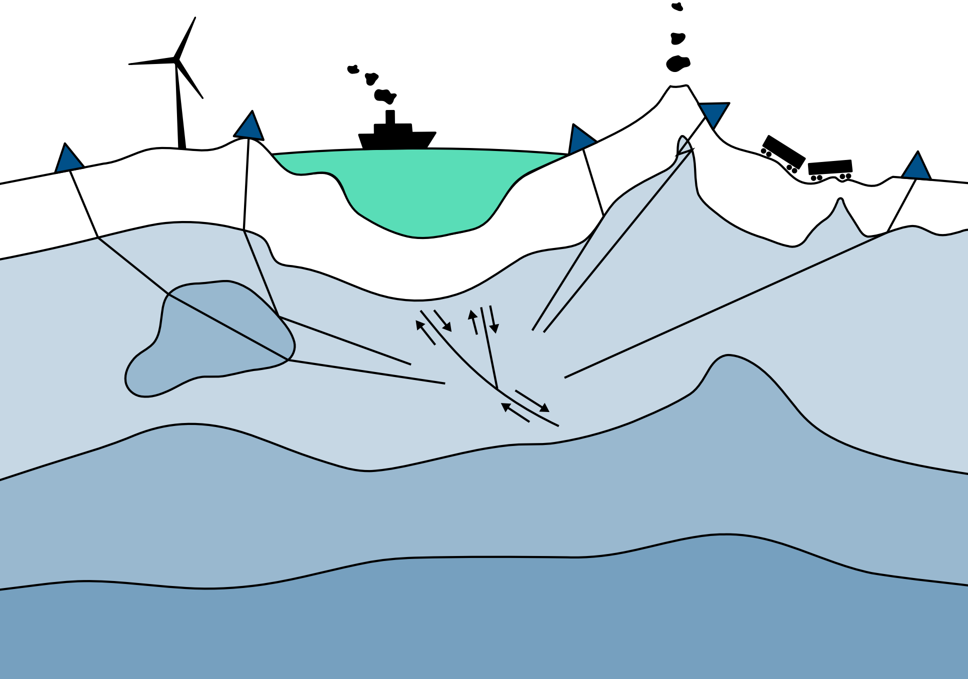


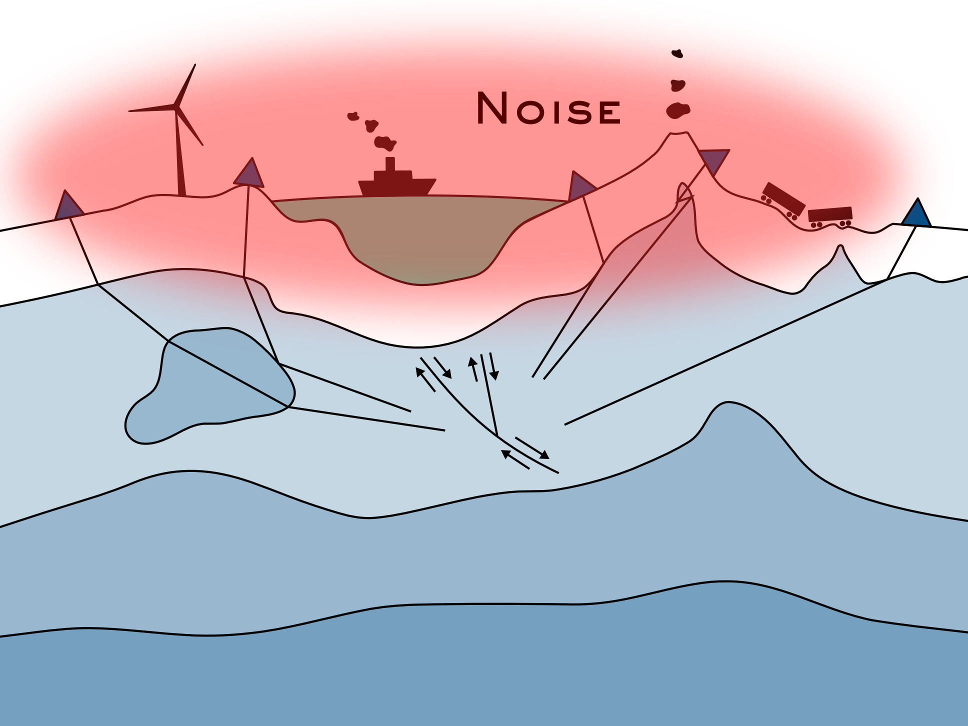


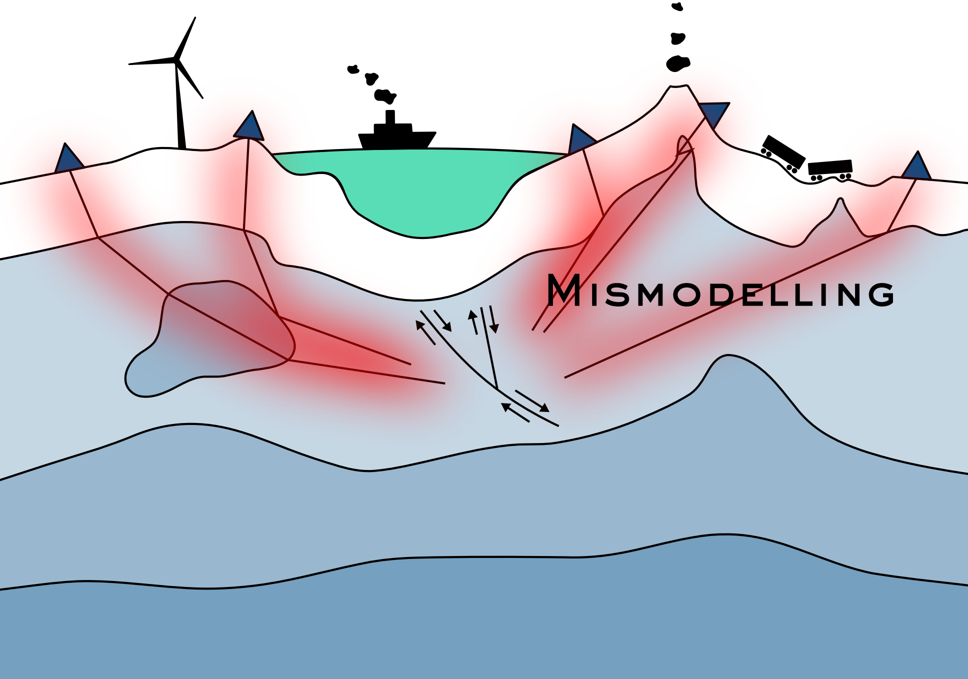


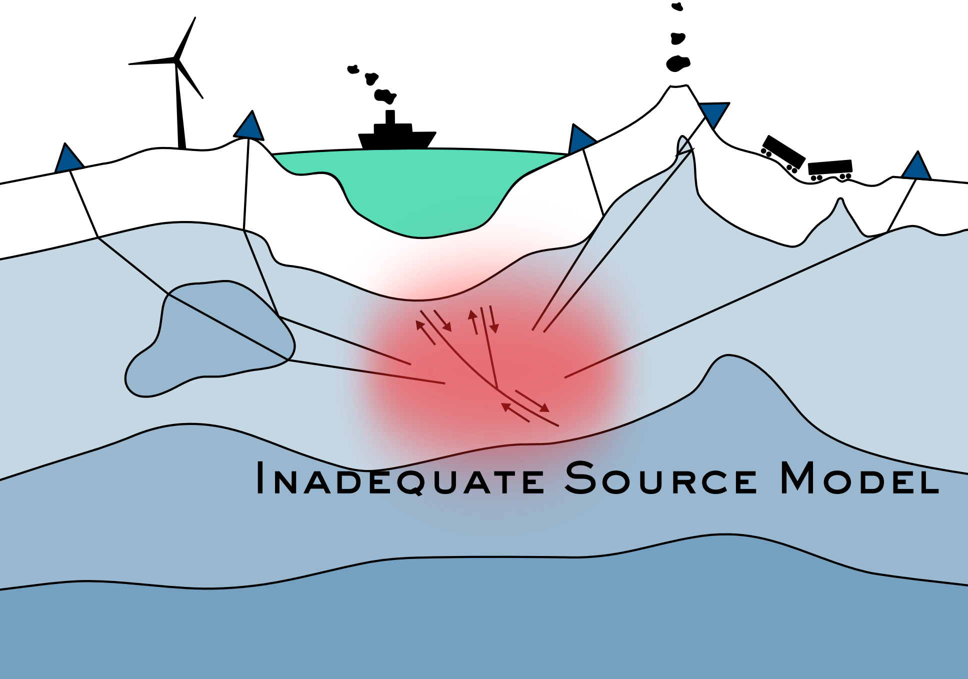







NOISE


Mismodelling

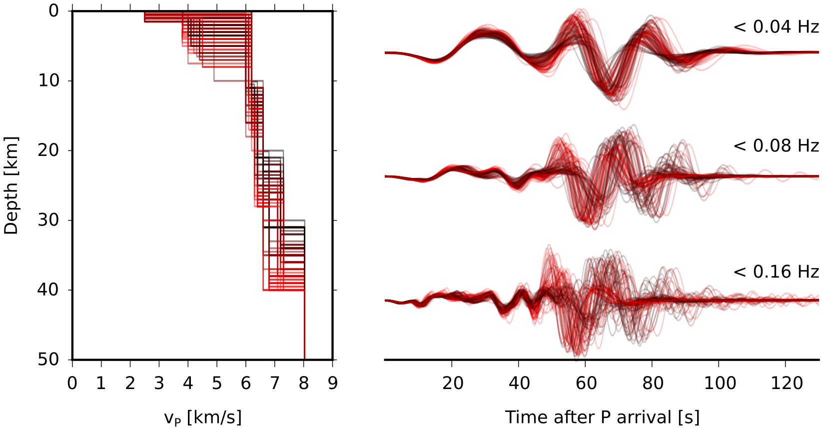

Inadequate Source Model

Seismology cannot answer everything!

- Where are the limits?
- What can be resolved?
- What uncertainties have to be expected?

Synthetic tests can help answer such questions!

- But synthetic tests can be a lot of work
- A fast and easy to use framework is needed


0
10
20
30
40
50
Depth [km]
0 1 2 3 4 5 6 7 8 9
v_P [km/s]
< 0.04 Hz
< 0.08 Hz
< 0.16 Hz
20 40 60 80 100 120
Time after P arrival [s]

Pyrocko/GF: API use example

```
from pyrocko import gf
km = 1000.

engine = gf.get_engine()

# my CRUST2.0 GF stores are named 'crust2_*'
store_ids = [x for x in engine.get_store_ids() if x.startswith('crust2_')]

# setup source and targets
source = gf.DCSource(
    lat=0., lon=0., depth=20.*km,
    strike=50., dip=40., rake=80.)

targets = [
    gf.Target(
        quantity='displacement',
        codes=('', 'STA', '', 'Z'),
        lat=2.5, lon=0.,
        store_id=store_id)

    for store_id in store_ids]

# calculate seismograms
resp = engine.process(source, targets)

# ... continued on next slide ...
```

Pyrocko/GF: API use example

```
# ... continued ...

for (source, target, tr) in resp.iter_results():

    # get moho depth for plot color
    store = engine.get_store(target.store_id)
    earthmodel = store.config.earthmodel_1d
    moho_depth = earthmodel.discontinuity('moho').z

    # align traces by P arrival
    tp = store.t('any_P', source, target)
    tr.shift(-tp)

    # filter and plot trace
    for i, fmax in enumerate((0.04, 0.08, 0.16)):
        trf = tr.copy()
        trf.lowpass(4, fmax)
        # plot...

    # plot P-wave velocities
    z = earthmodel.profile('z')
    vp = earthmodel.profile('vp')
    # plot...
```

Green's functions everywhere!

Many seismological methods require GFs

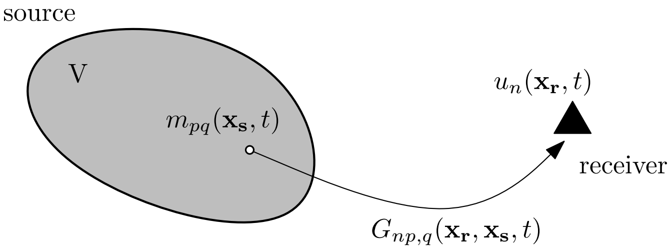
- Earthquake source inversion (MT, FS, STF, ...)
- Earthquake location / source imaging
- Array techniques
- Instrument performance evaluation
- Shake map generation
- Tsunami modelling
- Synthetic dataset generation

Variability

Many different GF setups and types are used

- global/regional/local/lab scale
- static dislocation, near-field, far-field
- acoustic, visco-elastic, poro-elastic
- full waveform, specific phases

Synthetic seismogram generation

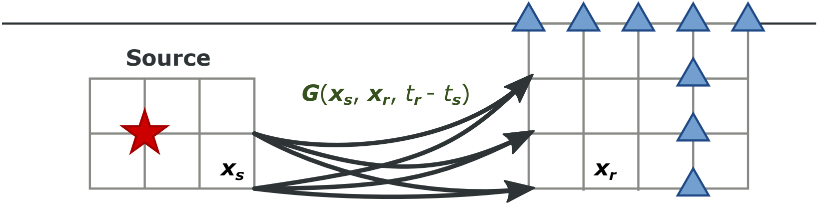


$$u_n(\mathbf{x_r}, t) = \iiint\limits_V m_{pq}(\mathbf{x_s}, t) * G_{np,q}(\mathbf{x_r}, \mathbf{x_s}, t)\,\mathrm{d}V \qquad \mathbf{x_s} \in V$$

- Compute as a sum of weighted and delayed GFs
- Use precomputed GFs

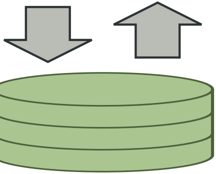
Synthetic seismogram generation

Observable quantity:
static $\boldsymbol{u}$, $\boldsymbol{u}(t)$, $\boldsymbol{v}(t)$, $p(t)$, ...



Numerical forward modelling:

- expensive
- expert task



Extract / recombine:

- cheap
- independent

Precomputed Green's functions

Precomputed Green's functions

Computation and storage of dense Green's function

$$G_{np,q}(\mathbf{x_r}, \mathbf{x_s}, t)$$

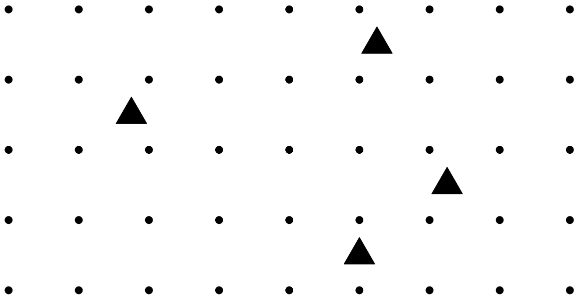
often not feasible.

Required storage size grows like

$$S \propto (f_{\max})^7.$$

Reduce storage cost

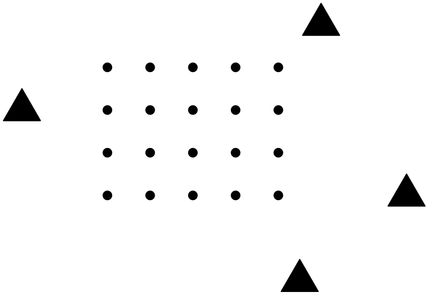
- evaluate only for sparse receiver positions $\mathbf{x_r}$



$$S \propto (f_{\max})^4$$

Reduce storage cost

- evaluate only for limited source region $\mathbf{x_s}$



$$S \propto (f_{\max})^4$$

Reduce storage cost

- restrict to layered earth model (use translational and rotational invariance)

$$G(\mathbf{x_r}, \mathbf{x_s}, t) \rightarrow G(z_r, z_s, \Delta, t)$$

$$S \propto (f_{\max})^4$$

- restrict to surface receivers

$$S \propto (f_{\max})^3$$

Green's function store

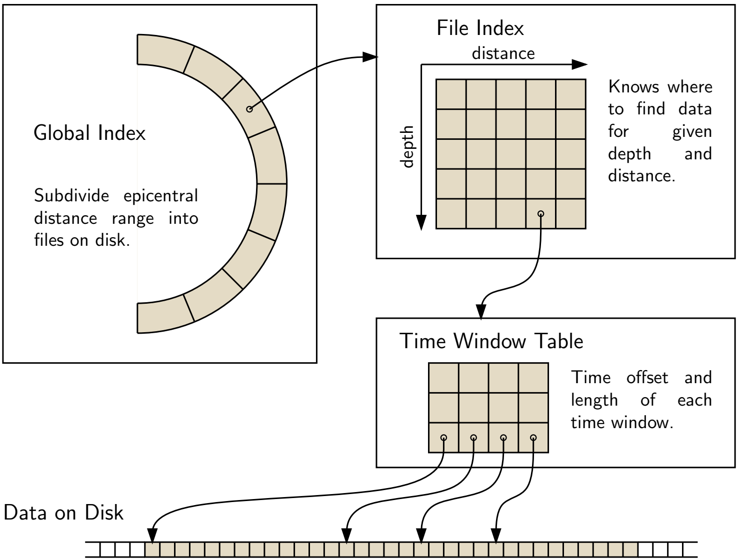
Requirements

- An efficient storage scheme is needed
- Reading of individual traces must be fast
  - write once
  - read often
  - random access

GF storage frameworks

- Kiwi Tools GFDB format (2005)
  `http://kinherd.org/`
- Pyrocko GF Store (2013) *new!*
  `http://pyrocko.org/`

Kiwi Tools GFDB



Kiwi Tools GFDB

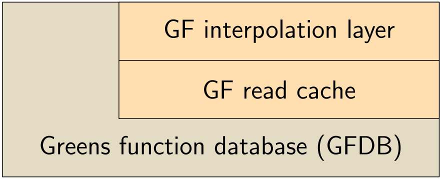
Implemented as a Fortran 95 module.

Features:

- easy to use interface
- binary and platform independent storage format (HDF5)
- actually used traces are cached in RAM
- transparent, on-the-fly interpolation:
  - between grid nodes (non-aliased traces): *bilinear interpolation*
  - to increase virtual grid resolution (aliased traces): *Gülünay 2D and 3D Fourier domain trace interpolation*

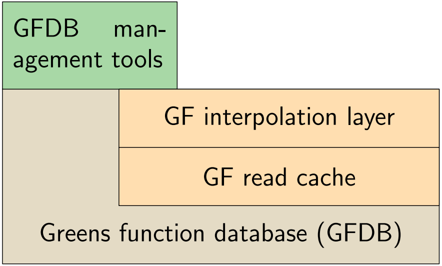
Kiwi Tools Software Architecture

The Kiwi Tools



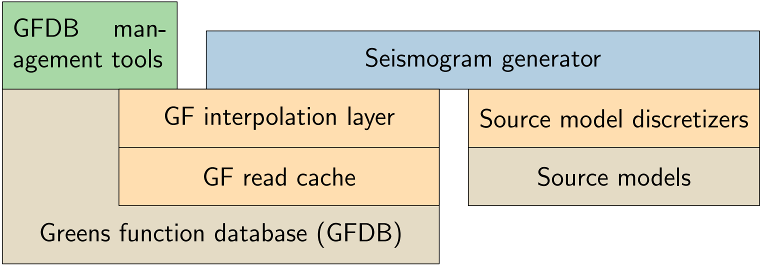
Kiwi Tools Software Architecture

The Kiwi Tools



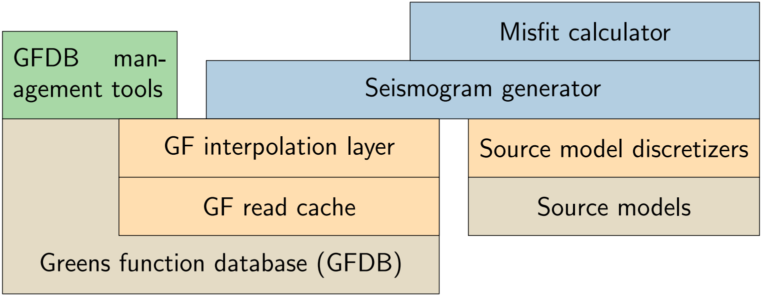
Kiwi Tools Software Architecture

The Kiwi Tools



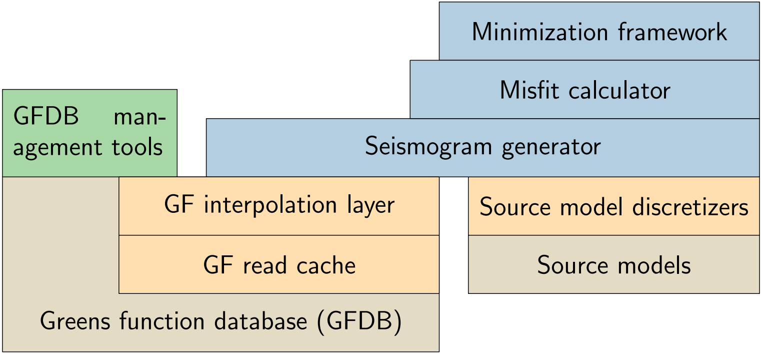
Kiwi Tools Software Architecture

The Kiwi Tools

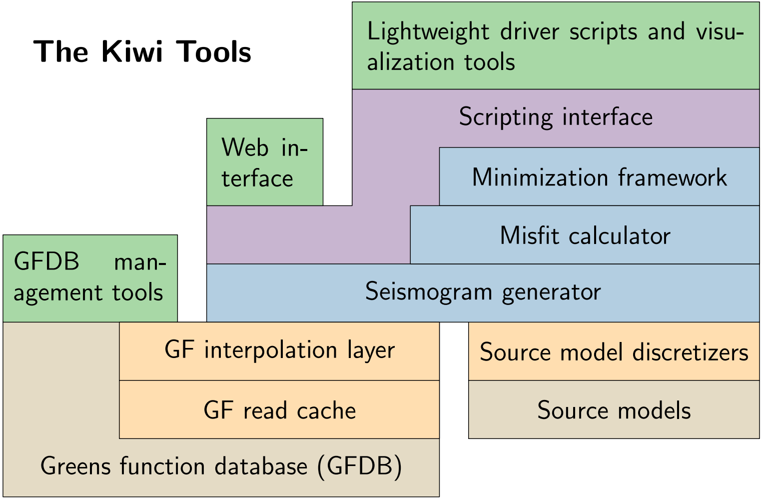


Kiwi Tools Software Architecture

The Kiwi Tools



Kiwi Tools Software Architecture



Nice

- Fast forward modelling
- Flexible
- Independent of GF modelling code
- Several precomputed GFDBs online:
  `http://kinherd.org/avail.html`

Nice, but...

- Fortran95 + Python
- HDF5 + Fortran95
- Parallel support
- Prerequisites

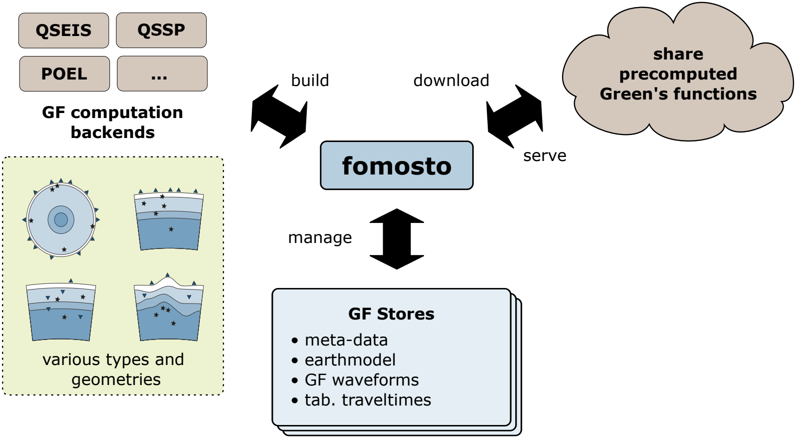
But it could be better...

- No metadata concept
- Strict (source-depth, surface-distance) indexing
- No integrated concept how to create GFDBs
- No integrated concept how to create travel time tables

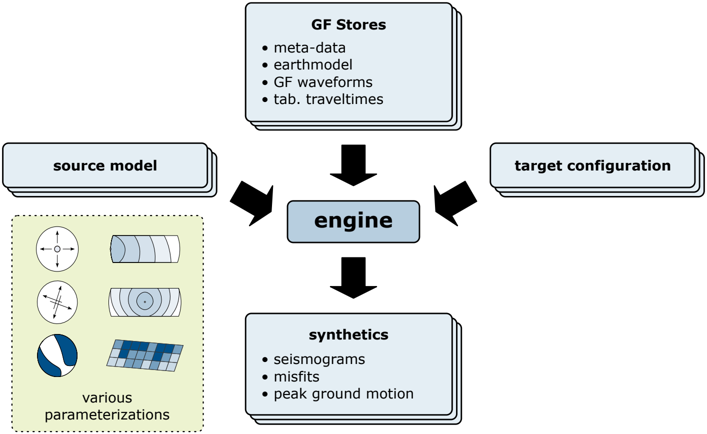
The new Pyrocko GF Store format

- Flexible indexing
  - Type A: 1D earth model, fixed receiver depth
  - Type B: 1D earth model, variable receiver depth
  - Type C: 3D earth model, cartesian CS for source region, fixed receiver location
- Simple, dependency free storage format
- Data model for GF Store meta data

The Pyrocko GF Fomosto Tool



The Pyrocko GF Seismosizer Engine



How to create GF Stores

Unified GF generation/management tool: **fomosto**

- Integrated traveltime table generation
- Integrated visualization
- Import/export to Kiwi GFDB format
- Download GF Stores from server

Fomosto's GF computation backends

Currently available GF computation backends:

- QSEIS - for regional seismograms (by Rongjiang Wang)
- QSSP - for global seismograms (by Rongjiang Wang)
- POEL - for poroelastic GFs (by Rongjiang Wang)

Practical: GF Store management

Fomosto Tutorial

`http://pyrocko.org/current/fomosto.html`

Pre-calculated Pyrocko GF stores: *The Green's Mill*

`http://kinherd.org:8080/gfws/static/stores`

Example: download GF store from command line

```
fomosto download kinherd crust2_dd
```

Practical: synthetic seismograms

1. `cd $HOME/playground/data/gfz2012wdpw/prepared`
2. `snuffler --stations=stations.txt --event=event.txt *PALK* *KBL* *TATO* *LHMI*`
3. right-click → Panels → Seismosizer
4. click "Run"
5. Filter between 0.005 and 0.01 Hz
6. look at Z components only, first
7. right-click → select "Common Scale per Station"
8. right-click → select "... (Grouped by Location)"
9. right-click → deselect "Show Boxes"
10. right-click → select "Color Traces"
11. Try to manually fit the synthetics to the observations!

A minimal Python script to create synthetics

```python
from pyrocko import gf, util, io

km = 1000.

source = gf.DCSource(
    time=util.str_to_time('2016-08-24 10:34:55'),
    lat=20.92, lon=94.64, depth=91*km,
    strike=352., dip=88., rake=-86.,
    magnitude=6.7)

targets = [
    gf.Target(
        codes=('', 'NPT', '', comp),
        lat=19.75, lon=96.1,
        store_id='regional_2s',
        interpolation='multilinear')
    for comp in ['N', 'E', 'Z']
]

engine = gf.get_engine()
response = engine.process(source, targets)
io.save(response.pyrocko_traces(), 'out.mseed')
```