

Many geophysical methods require knowledge of Green's functions (GF) or synthetic seismograms in dependence of ranges of source and receiver coordinates. Examples include synthetic seismogram generation, moment tensor inversion, the modeling of depth phases for regional and teleseismic earthquakes, or the modeling of pressure diffusion induced static displacement and strain.

It can be of advantage to calculate the GFs in advance: the same GF traces can then be reused several or many times as required in a typical application. Regarding GF computation as an independent step in a use-case's processing chain encourages to store these in an application independent form. They can then be shared between different applications and they can also be passed to other researchers, e.g. via web service.

+ Public HTTP API (soon) at

<http://kinherd.org/>

+ Open source
+ Written in Python
+ Develop with us at

<http://emolch.github.com/pyrocko/>



Use in own script

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

km = 1000.
run_local = True

sources = [
    # use one simple double-couple source for demo
    gf.DCSource(
        time=util.str_to_time('2013-01-01 10:00:00'),
        lat=0.0,
        lon=0.0,
        depth=10*km,
        strike=45.,
        dip=90.0,
        rake=0.0,
        magnitude=6.0)
]

targets = [
    # use a single three component receiver
    gf.Target(
        codes=(' ', 'STA', ' ', 'cha'),
        lat=10.,
        lon=0.,

        # Green's functions: use global AK135 0.5 Hz GFs created with QSSP
        store_id='global_2s',

        # could request seismogram with a lower sampling rate (speedup)
        sample_rate=None,

        # could request any component orientation
        azimuth=None,
        dip=None,

        # could request to add some post-processing
        pre_taper=None,
        filter=None,
        post_taper=None)

    for cha in ['BHZ', 'BHE', 'BBN']]

if run_local:
    engine = gf.LocalEngine(store_superdirs=['/data2/heimann/gf_stores'])
else:
    engine = gf.RemoteEngine(site='kinherd')

# create synthetic seismograms
response = engine.process(sources=sources, targets=targets)
traces = response.pyrocko_traces()

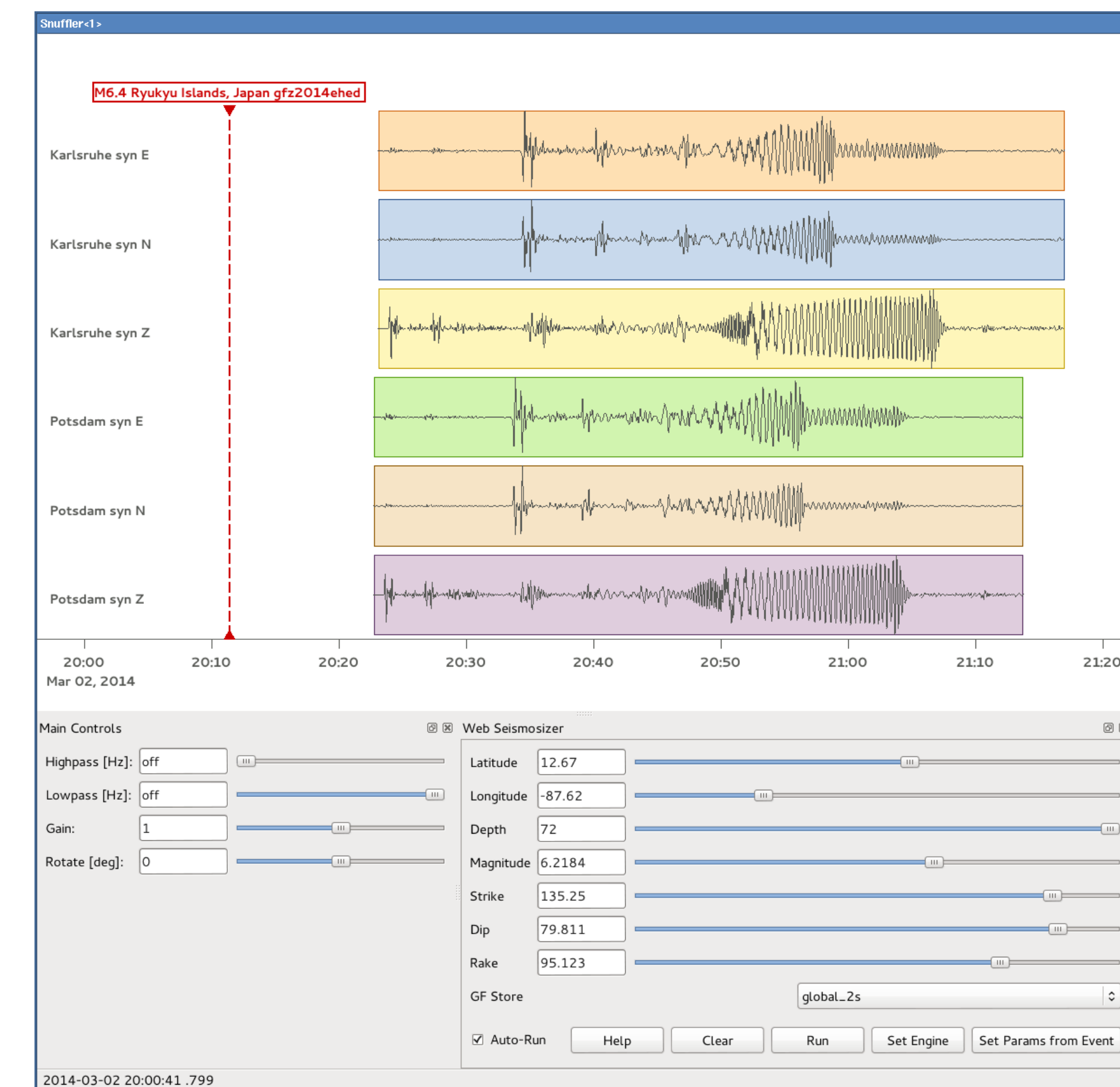
# save traces in MiniSEED format
io.save(traces, 'synthetics.mseed')

# quickly look at traces
trace.snuffle(traces)
```

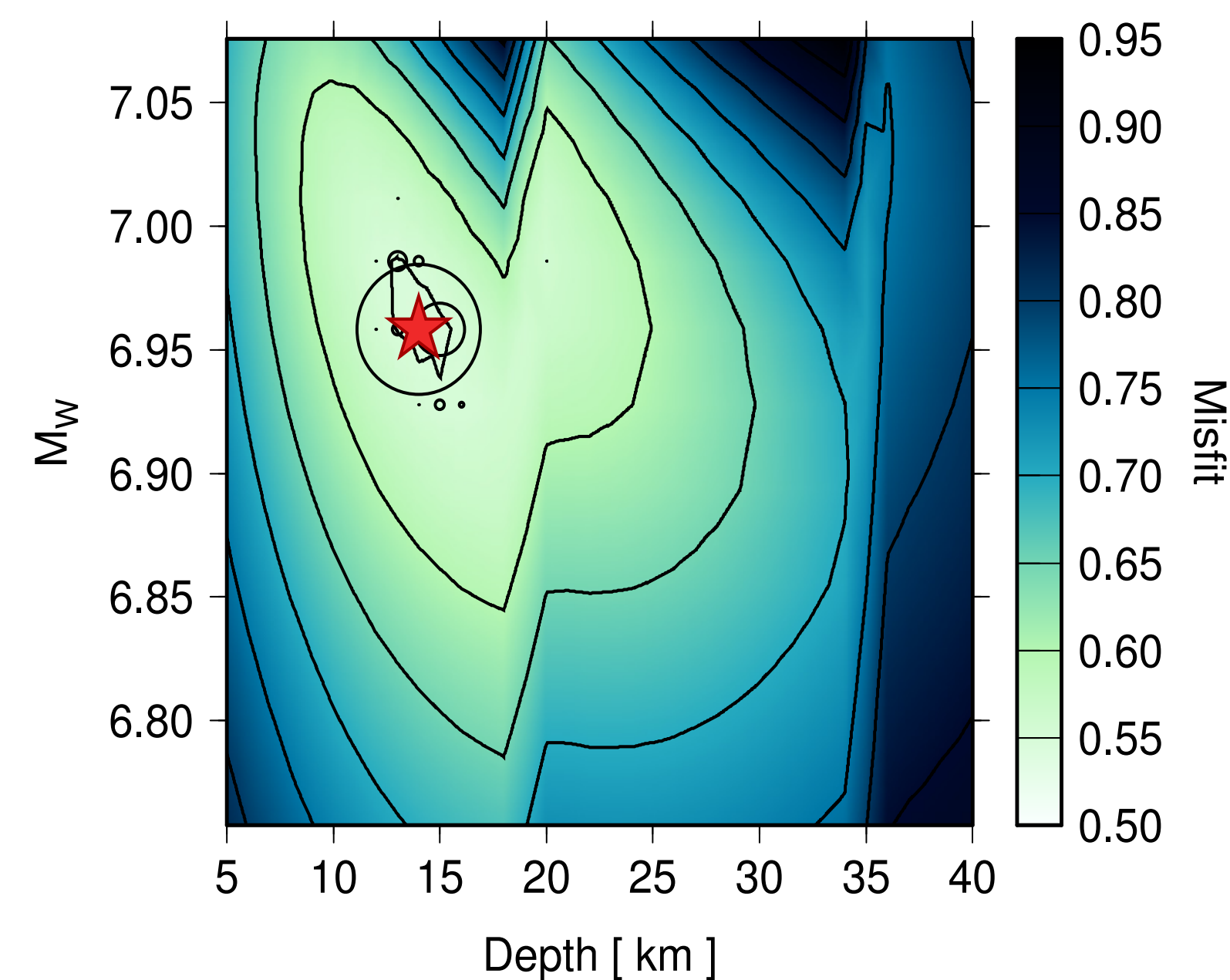
Synthetic Seismogram Web Service and Python Tools

Heimann, S.¹, Cesca, S.¹, Kriegerowski M.², Dahm, T.¹

Starting now, we provide such a web service to the seismological community at <http://kinherd.org/>, where a researcher can share GF stores and retrieve synthetic seismograms for various point and extended earthquake source models for many different earth models at local, regional and global scale. This web service is part of a rich new toolset for the creation and handling of GFs and synthetic seismograms. It can be used off-line or in client mode.



Use interactively



Quickly explore source-model parameter-space or feed misfit values into your favourite optimization algorithm!

The Pyrocko/GF Seismosizer

An egg-laying wool-milk-pig for seismogram synthesis

