pyNetOpt3D

A Python API for Monitoring Network Optimization

Tobias Megies (LMU Munich), Toni Kraft (ETH Zurich)

Schweizerischer Erdbebendienst
Service Sismologique Suisse
Servizio Sismico Svizzero
Swiss Seismological Service

Network Optimization for Geotechnical Projects

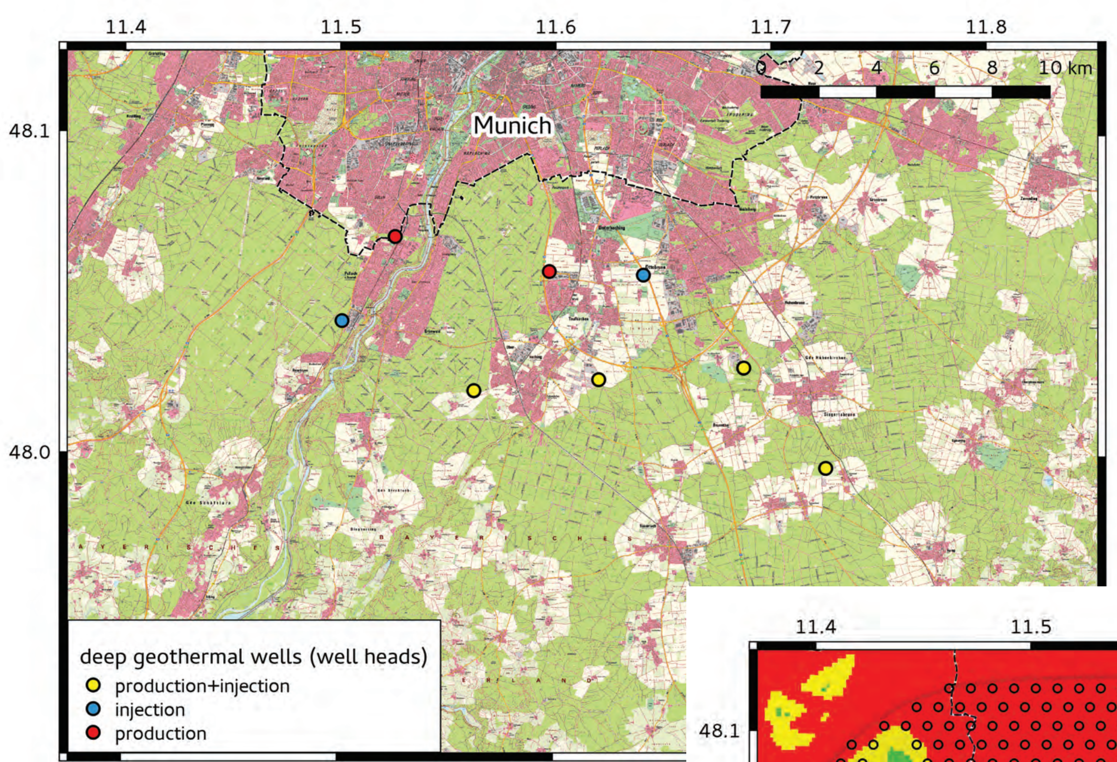
At the Swiss Seismological Service (SED), **monitoring network design for geotechnical projects that could cause induced seismicity** is conducted using a set of C codes (NetOpt3D) based on work by Hardt and Scherbaum (1994) and further developed by Kraft et al. (2013). **NetOpt3D finds D-optimal designs** using a simulated annealing approach and handles 3D velocity and attenuation models, accounts for existing stations and the expected SNR of bodywaves at existing and new stations based on models of seismic noise (Kraft 2016).

The Problem

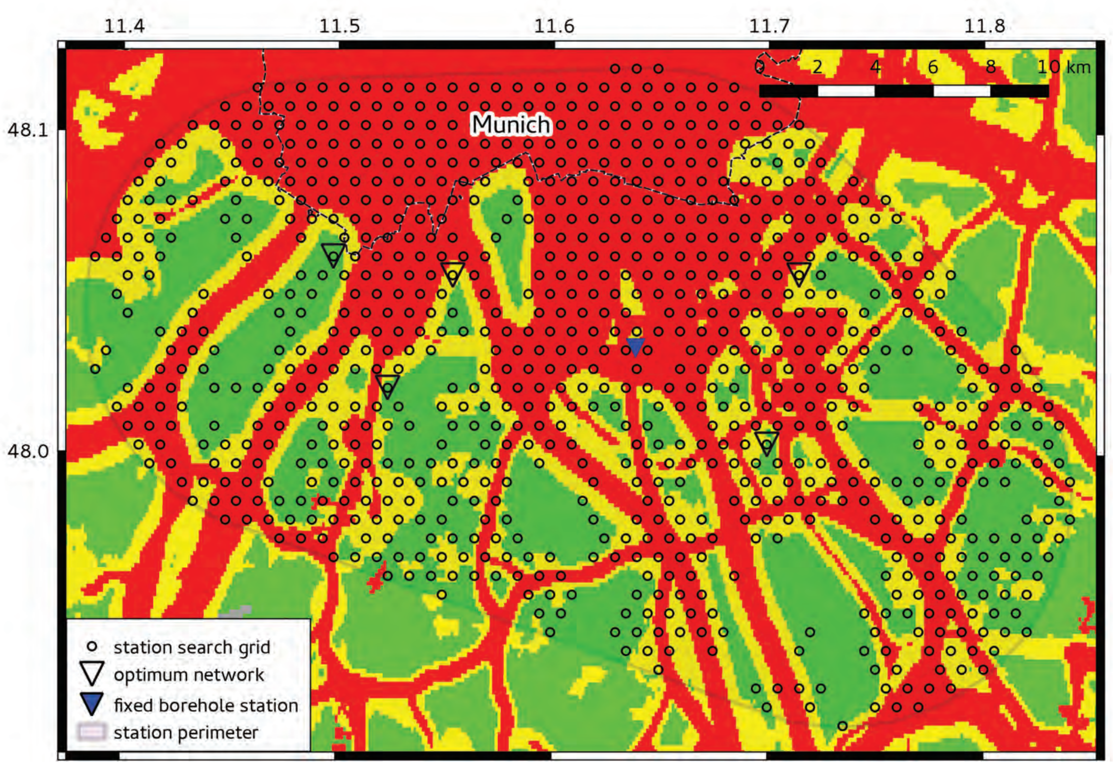
In its current form, **NetOpt3D is lacking usability and it is time consuming to set up new optimization problems**. Input files (velocity models, synthetic catalogs, ...) have to be set up manually in fixed legacy ASCII formats defined by the underlying C codes and a large number of helper programs (Linux shell scripts, ...) are used for preparational steps (e.g. to calculate convex hull of earthquakes, buffer it and set up equidistant station grids) and for analysis and visualization of results (mostly GMT scripts).

Our Solution

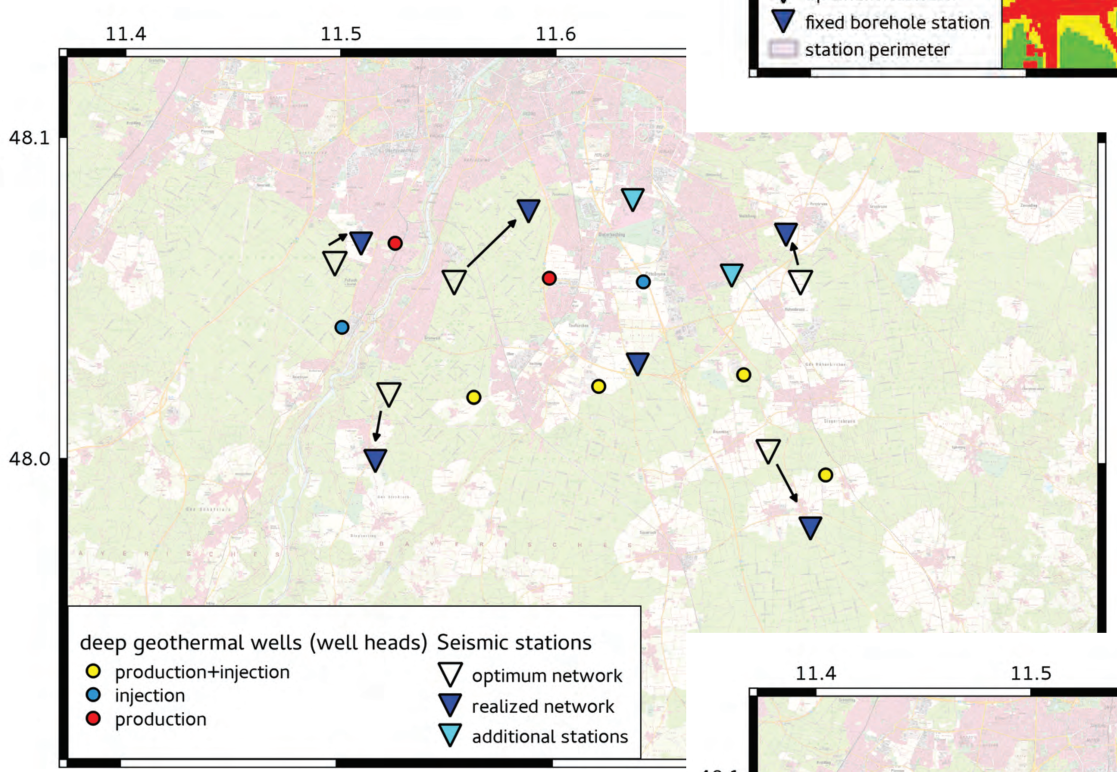
We are developing a **consistent and easy-to-use Python API pyNetOpt3D** that internally uses NetOpt3D C codes but hides all unwieldy steps from the user (input file setup, parsing of results, ...).
It will be possible to steer a complete optimization run with a single, short Python script using the newly developed API. We will also enhance reproducibility by providing (de)serialization of a full optimization run including all input data and results into a single file. Visualization of optimization re-



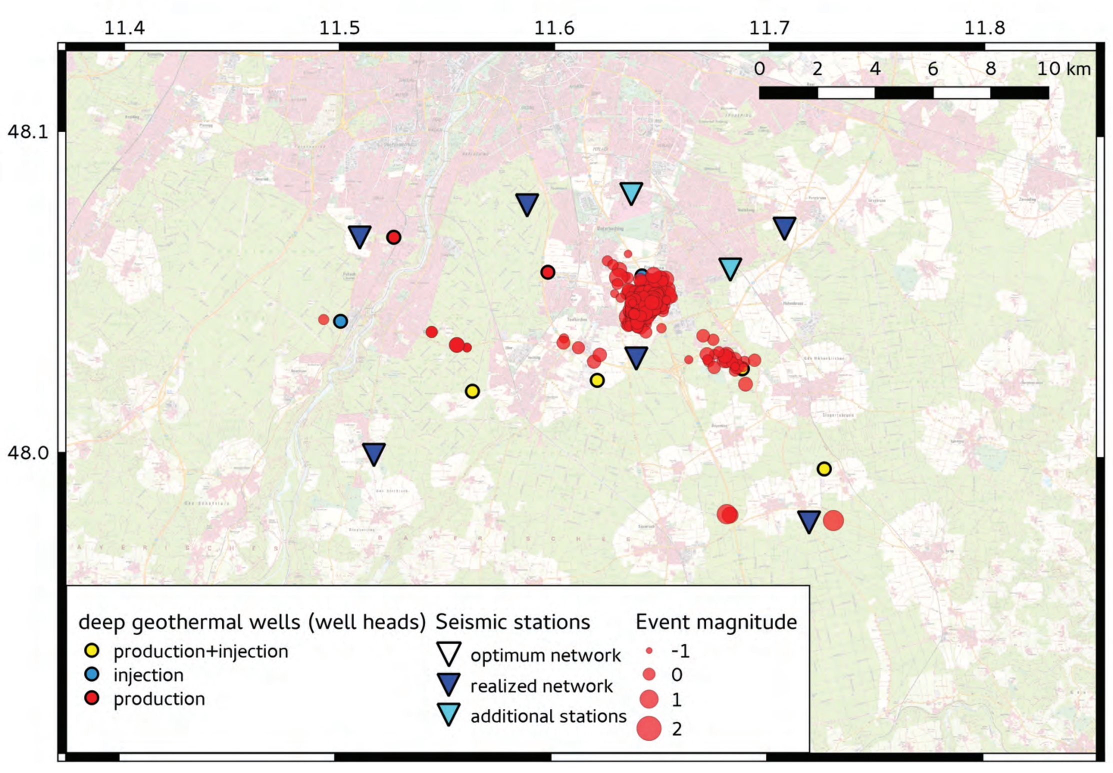
Left: Example use case for monitoring network optimization. Arrangement of six closely spaced deep geothermal doublets near Munich, Germany.



Right: Station perimeter around injection well open-hole sections as hypothetical event locations with equidistant station grid (excluding locations without power grid access). Noise model by Kraft (2016) was used to estimate noise levels at potential stations. Optimum network shown as hollow triangles.



Left: Final network realization after test measurements in several locations for each station.



Bottom: Microseismicity observed by the network. Events with magnitudes below zero could be observed despite the low number of stations owed to the diligent network design.

References:

Hardt, M., & Scherbaum, F. (1994). *The design of optimum networks for aftershock recordings.* Geophysical Jour-nal International, 117(3), 716-726.

Kraft, T., Mignan, A., & Giardini, D. (2013). *Optimization of a large-scale microseismic monitoring network in northern Switzerland.* Geophysical Journal International, 195(1), 474-490.

Kraft, T., *A high-resolution and calibrated model of man-made seismic noise for Europe*, 76. DGG annual meet-ing, Münster, 16. March 2016

Main Features of pyNetOpt3D

Easy-to-use Python API around NetOpt3D

All functionality of pyNetOpt3D is available via a clean Python API, making it possible to steer a complete optimization problem using a single, short Python script. NetOpt3D C codes are used under the hood.

Automatic Coordinate Conversions, Convex Hulls, Buffers, Station Grids

All coordinate conversions from global geographic coordinates (WGS84) to local geodetic coordinates (e.g. UTM, Gauß-Krüger, Swiss Grid, ...) and vice versa are handled automatically (using pyroj/proj.4). Functionality to calculate convex hulls, buffers and equistant station grids is included.

(De)serialization

The complete optimization run including all input parameters (velocity model, raytracer parameters/output, optimization parameters/output, ...) can be easily serialized to a single file. Results are therefore fully reproducible at a later time and can easily be loaded/plotted again.

Convenience Command Line Tools

pyNetOpt3D provides command line tools to quickly print/plot an optimization run from a serialized file on disk.

Outlook

* pyNetOpt3D will be published within the coming 2-3 months with pre-compiled NetOpt3D binaries
* NetOpt3D C codes are currently being rewritten in an effort to clean up the code, enhance maintainability, add new features (e.g. borehole arrays) and solve licensing issues to enable an eventual open source release

```
from pyNetOpt3D.core import NetOpt
from pyNetOpt3D.models import VelocityModel1D, ScalarQModel
from pyNetOpt3D.parameters import (
    OptimizationParameters, RaytracerParameters)

# setup velocity model
depths = (-0.4, 0.5, 2.2, 3.0, 14.7, 15.3, 32.7, 33.3, 100.0)
vp = (2.7, 3.5, 4.5, 6.0, 6.0, 6.5, 6.5, 8.1, 8.1)
vpvs = (1.86, 1.86, 1.86, 1.73, 1.73, 1.73, 1.73, 1.73, 1.73)
v_mod = VelocityModel1D(z=depths, vp=vp, vpvs=vpvs)

# setup Q model
q_mod = ScalarQModel(
    kappa_p=0.007, kappa_s=0.016, q_p=2736, q_s=1216)

net_opt = NetOpt(velocity_model=v_mod, q_model=q_mod)

# setup earthquake data, place below well heads in this example
wells = [
    ('Unterhaching', 11.64121, 48.05612, 3.0),
    ('Oberhaching', 11.56268, 48.01986, 3.0),
    ('Pullach', 11.50123, 48.04140, 3.0),
    ('Taufkirchen', 11.62067, 48.02347, 3.0),
    ('Kirchstockach', 11.68795, 48.02732, 3.0),
    ('Duerrnhaar', 11.72620, 47.99629, 3.0)]
for name, lon, lat, depth in wells:
    net_opt.add_event(lon, lat, depth, magnitude=1.5, fps=None)

# setup station perimeter and grid, assign noise levels
net_opt.set_station_perimeter_from_events(buffer_in_km=8)
net_opt.set_potential_stations_from_station_perimeter(
    grid_spacing_in_meters=1100, grid_type='equidistant_triangular')
net_opt.set_potential_stations_noise_from_npy('noise_munich.npy')

# optionally, adjust raytracer parameters
net_opt.raytracer_parameters = RaytracerParameters()
net_opt.run_raytracing()

# set optimization parameters (annealing schedule, ...)
opt_params = OptimizationParameters(number_of_new_stations=6)
net_opt.optimization_parameters = opt_params
net_opt.run_optimization()

# serialize optimization (for later reuse) and plot
net_opt.save('example.netopt.npz')
net_opt.plot(filename='example.png')
```

This project is funded by
swiss energy
in the framework of project
GEOBEST-CH