nmPy Manual

Simon Schneider

September 2018

Contents

1 Introduction

What it is all about

2 Run Synthetics

This section describes how to use nmPy to calculate synthetic data using mdcpl, matdiag and synseis.

```
from nmpy.core.modes import read as read_modes
from nmpy.SetupRuns import settings
from nmpy.SetupRuns.setup import build_synthetics

args = settings.default_config()
args['intype'] = 'synthetics'
args['inmodel'] = 'S20RTS'
```

```
args['rundir'] =
    '//nfs/stig/simons/splitting/tests/synthetics/00s10-00t11/Z'
args['confdir'] = args['rundir']
args['modes'] = read_modes().select(name='0S10')
args['modes'] += read_modes().select(name='0T11')
args['cross_coupling'] = 1
args['cst_max'] = 12
args['cmt'] = ['060994A']
# Alternative datadir:
args['datadir'] = '//nfs/stig/simons/alldata/VHZ/'

setup = build_synthetics(args, remove_existing=True, run_code=True)
```

3 Get Data

3.1 Download

3.2 Cleaning and Conversion

```
remtidah = '~/bin/remtidah'

from nmpy.util.data_request import process_downloader_data
from nmpy.util.read import read_std_cat

inv = None
cat = read_std_cat('011012A')
components = ["N", "E", "Z"]
sampling_rate = 0.1
localfolder = '/data/simons/3-comp-data/120hrs/VH_raw'
inspected_only = False
remove_deltas = True
rotate_traces = True
rm_response = True
rm_tidal = True
keep_longest_traces = True
cut_to_same_length = False

process_downloader_data(inv, cat, components, sampling_rate, localfolder,
                        inspected_only,
                        remove_deltas,
                        rotate_traces,
                        rm_response,
                        rm_tidal,
                        keep_longest_traces,
                        cut_to_same_length)
```

3.3 Select Data Sets

```
from nmpy.preprocessing.data_correction import select_station
st = select_station('file.ahx', tw=[5, 60], fw=[1, 4], min_meansnr=1.2)
```

Will loop over all noise windows defined in `nmpy/data/AD/noisewindows.dat`. Sets the window between 2 noisewindows as signalwindow, calculates the maximum in this window. Does the same with the noise in the 2 neighbouring windows, picks the biggest peak in these. The SNR is defined as the ratio $\frac{max(Signal)}{min(Noise)}$ of those values. The code repeats that for all signal windows in the given frequency interval `fw=[frequency 1, frequency 2]` and calulates the average of the SNR's. If it is smaller then `min_meansnr` the station will be deleted.

Each new event

- calculate 0-2 mHz broadband
- look at whole spectrum
- delete bad stations permanently
- look for globally evenly distributed station pattern
- Check for events that occurred after the earthquake of interest (1.5-2 Mw difference)

Stations

- remove (boxcar/delta pulse) glitches automatically
- mark maximum tw per station and per event across all stations

Picking

- loop over freq bands (0-1, 1-2, 3-4,... mHz)
- define stations according to these bands
- have a list of tw for each band
- pick fw and tw (blindly)
- Set tw_{end} at first to qcycle length
- later try to set it bigger, if it still makes sense
- look at snr and m, which will depend on the mode type (mantle, IC, ...)
- select and run inversion
- Delete outlier stations (histogram of misfits), and run inversion again

It will be more important to have more events picked, than to have more segments of the same event For this, spectrum can be used, by running it with e.g.:

```
data = '060994A.ahx'
spectrum(data=data, fw=[0.3, 2], tw=[5, 60], minispec=True)
```

Picking for FSI Start with Q-cycle and extend time windows later Choose between multiple modes in one segment or cutting to one mode per segment If multiple modes in one segment: use multiple time windows in which the different modes are dominant If one mode (maybe two) per segment: use Q-cycle at first. Maybe downsample to 100 seconds for low frequency range (e.g. 0-3 mHz)

4 Spectrum

4.1 Plotting

4.2 Pick Segments

5 CST/FSI Inversion

5.1 Build CST Inversion

```
from nmpy.SetupRuns.damping_test import run_inversion
args = default_config()
allmodes = read_modes()
motherdir = '//nfs/stig/simons/splitting/tests'

run = 'ANTO'
args['comp'] = ['T', 'Z']
args['number_of_tasks'] = 10
args['cst_max'] = 4
args['cross_coupling'] = 1
args['autodatadir'] =
    '//nfs/stig/simons/splitting/tests/synthetics/00s10-00t11'

Modes = [['0T11', '0S10']]
damping = np.logspace(-4, 2, 7)[2:]
for modes in Modes:
   run_inversion(modes, args, damping, run, motherdir)
```

If you want to start the inversion from an existing file just change `args['modeldir']` to the path where the file is saved. This file has to be the same format of course.

5.2 Analyse Inversion Results