"OpenQuake: Calculate, share, explore"

OpenQuake Ground Motion Tookit - User Guide

PUBLISHED BY GEM FOUNDATION

GLOBALQUAKEMODEL.ORG/OPENQUAKE

Citation

Please cite this document as:
Weatherill, G. A., Pagani, M., Monelli, D. and Garcia, J. (2014) OpenQuake Ground Motion Toolkit - User Guide. *Global Earthquake Model (GEM). Technical Report*

Disclaimer

The “OpenQuake Ground Motion Toolkit - User Guide” is distributed in the hope that it will be useful, but without any warranty: without even the implied warranty of merchantability or fitness for a particular purpose. While every precaution has been taken in the preparation of this document, in no event shall the authors of the manual and the GEM Foundation be liable to any party for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of information contained in this document or from the use of programs and source code that may accompany it, even if the authors and GEM Foundation have been advised of the possibility of such damage. The Book provided hereunder is on as "as is" basis, and the authors and GEM Foundation have no obligations to provide maintenance, support, updates, enhancements, or modifications.

eeds to be up-
or removed

The current version of the book has been revised only by members of the GEM model facility and it must be considered a draft copy.

License

First printing, June 2014

Contents

1. Introduction

1.1 A Ground Motion Prediction Equation (GMPE) Toolkit

Probabilistic Seismic Hazard Assessment (PSHA) is well-established as a robust and effective means of characterising the level of ground shaking to which a structure may be subject within a given time period. Since its inception in the 1960s the PSHA methodology undergone many developments designed to improve the characterisation of the physical properties of the rupture source, the ground motion attenuation and the site amplification. Often leading these developments are the Ground Motion Prediction Equations (GMPEs), which characterise the manner in which ground motion measures (intensity measures) vary with the source, path and site properties of observed ground motion records.

The widespread adoption of PSHA for site-specific, urban-level and national/regional-scale analyses of earthquake hazard has created a clear need for calculation software to be widely available for this purpose. Furthermore, in recent years that has been an increasing need for PSHA calculation software to be widely available, for its processes to be open, and for a commitment to quality assurance. Recognising this need the Global Earthquake Model (GEM) Foundation created OpenQuake, an open-source software for calculation of probabilistic seismic hazard and risk, that is developed using a test-driven philosophy in which the software undergoes extensive testing and quality assurance as part of the construction process.

Whilst OpenQuake itself is designed to fullfill the need for open and tested software for PSHA calculation, it does not explicitly address the issue of PSHA model construction. A PSHA model can be broken down into three components: the seismogenic source model(s), the ground motion prediction equations and the logic tree to characterise the model-to-model uncertainty in an equation. GEM is now in the process of endeavouring to extend the OpenQuake development approach to the area of PSHA model construction. It has done so with the development of two "Modeller's Toolkit's:

1. **The Hazard Modeller's Toolkit** , a suite of open-source tools for the preparation of seismogenic source models for application in PSHA citeWeatherill2014
2. **The OpenQuake Ground Motion Toolkit**, a suite of open-source tools for analysis and interpretation of observed ground motions and ground motion prediction equations, for the purposes of GMPE selection in PSHA (this document)

This document presents the OpenQuake Ground Motion Toolkit, or GMPE-SMTK hereafter, providing an overview of the features as well as practical examples that demonstrate how to use

the toolkit in real applications.

1.2 Objectives and Structure

The GMPE-SMTK serves the primary objective of assisting seismic hazard modellers with the process of understanding and identifying GMPEs for application in seismic hazard analysis. It brings together a collection of tools for visualising and exploring ground motion prediction equations, for analysing ground motion records to derive widely used ground motion intensity measures, and to compare observed ground motion intensity levels against the values predicted by the GMPE. To do this, however, the GMPE-SMTK has needed to create an internal data structure to store and process databases of ground motion records. A general overview of the GMPE-SMTK is shown in Figure 1.1.

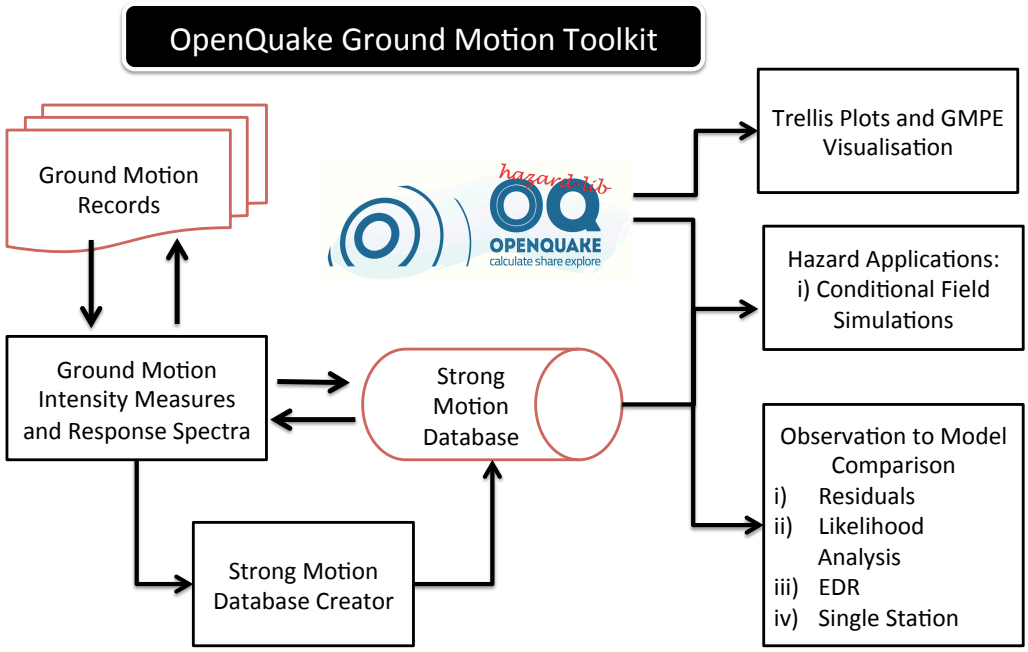


Figure 1.1 – *Feature Overview of the OpenQuake Ground Motion Toolkit*

The critical element within this process is the usage of OpenQuake, or more specifically the OpenQuake Hazard Library, as a dependency within the toolkit. It is from this library that we take the GMPEs, as well as various tools for undertaking calculations of source to site geometry. By using OpenQuake's own library we inherit OpenQuake's testing and quality assurance process for implementation of GMPEs. We also ensure total consistency between the GMPEs used for comparison against strong motion records and those used within the OpenQuake PSHA calculation engine itself. This is a feature that is unique to this toolkit.

A list of the current functionalities is given in Table 1.1.

1.3 Installation and Getting Started

The software is available on the open Github repository: https://github.com/g-weatherill/gmpe-smtk. Installation is dependent on the OpenQuake hazardlib and nrmllib libraries. These can be retrieve from the following repositories:

Feature	Algorithm
Ground Motion Intensity Measures	Retrieve PGA, PGV and PGD Response Spectra - Newmark-β Response Spectra - citeNigam Jennings1969 Horizontal Spectra (Geometric Mean, Envelope etc.) Rotate Horizontal Record Pairs Rotational Spectra (GMRotD, GMRotI) citeBooreetal1996
Trellis Plotting	Configure Rupture Compare GMPE Scaling with Magnitude Compare GMPE Scaling with Distance Compare GMPE Spectra with Magnitude & Distance
Comparison with Observations	Get GMPE residuals from Ground Motion ... ... measure residual distribution ... retrieve inter- and intra-event residual ... view trends with magnitude ... view trends with distance ... view trends with V_{S30} ... view trends with event depth Likelihood Analysis citeScherbaum et al. 2004 Log-likeihood Analysis citeScherbaum etal. 2009 Euclidean Distance-Based Ranking (Cite KaleAkkar2013) Single-Site Residual Analysis
Hazard Applications	Conditional Field Simulation

Table 1.1 – *Current features within the OpenQuake Ground Motion Toolkit*

- OpenQuake Hazard Library (oq-hazardlib)
  https://github.com/gem/oq-hazardlib
- OpenQuake Natural Risk Markup Language (NRML) Library (oq-nrmllib)
  https://github.com/gem/oq-nrmllib

The GMPE-SMTK has the following dependencies:

- **Numpy 1.6.1 or later**
- **Scipy 0.11.0 or later**
- **Matplotlib 1.3.x or later**
- **H5py 2.2.0 or later**
- **lxml**
- **Shapely 1.2.18 or later**
- **oq-hazardlib (brings in Numpy, Scipy and Shapely)**
- **oq-nrmllib (brings in lxml[1])**

1.3.1 Windows

For a comprehensive Python installation including many of the dependencies needed to both the toolkit and the OpenQuake libraries the best option is a full installation of PythonXY. This software is freely available from here: https://code.google.com/p/pythonxy/. It is recommended to proceed with a *full* installation and ensure that all of the above dependencies are selected for installation where available.

For the oq-nrmllib it is necessary to install the Lxml library (http://lxml.de). This is not officially supported for Windows, so it is recommended (by Python developers themselves!) to install the unofficial Lxml binding from here: [2]. Download and then run the 32-bit package named "lxml-#.#.#.win32-py2.7.exe".

Next you will need to install the op-hazardlib and oq-nrmllib. From the web repositories listed previously click the button "Download Zip", then extract contents to the folders `C:/oq-hazardlib` and `C:/oq-nrmllib` respectively.

Open an enhanced IPython console. Go to Start $->$ Python(xy) $->$ Enhanced Consoles $->$ Ipython (sh). This will open up an Ipython console terminal. To install the oq-hazardlib, at the console prompt type:

```
~$: cd C:/oq-hazardlib/
~$: python setup.py install build --compiler=mingw32
```

This will install the oq-hazardlib with the full C-extensions, which speed up some of the geometry calculations. Then do the same with the oq-nrmllib.

```
~$: cd C:/oq-hazardlib/
~$: python setup.py install
```

Then close the console by typing `exit`.

Finally, download the zipped folder of the GMPE-SMTK from the github repository and unzip to a folder of your choosing. To allow for usage of the GMPE-SMTK throughout your operating system, do the following:

1. From the desktop, right-click **My Computer** and open **Properties**
2. In the "System Properties" window click on the **Advanced** tab.
3. From the "Advanced" section open the **Environment Variables**.

[1] Except on Windows

[2] http://www.lfd.uci.edu/ gohlke/pythonlibs/

4. In the "Environment Variables" you sill see a list of "System Variables", select "Path" and "Edit".
5. Add the path to the hmtk directory to the list of folders then save.

After this process it may be necessary to restart PythonXY.

1.3.2 OSX and Linux

For installation on unix platforms we recommend installing each package using the Python Package manager (Pip). Assuming that Python 2.7.3 is already installed on your system, the Package Manager can be installed using the command line:

```
~$ python get-pip.py
```

First it is recommended to install the OpenQuake libraries (as they bring in most of the dependencies!). These can be done using Git to clone the repositories:

```
# Clone the hazardlib repository
git clone https://github.com/gem/oq-hazardlib.git
cd oq-hazardlib
# Run the installation
sudo python setup.py install
cd ..
# Clone the nrmllib repository
git clone https://github.com/gem/oq-nrmllib.git
cd oq-nrmllib
sudo python setup.py install
```

Then use PiP to install Matplotlib and H5py:

```
# Install Matplotlib
sudo pip install matplotlib
# Install H5py
sudo pip install h5py
```

Finally all that is needed is to install the GMPE-SMTK. As with the OpenQuake libraries the best option is to use Git to clone the repository:

```
# Clone the repository
~$ git clone https://github.com/g-weatherill/gmpe-smtk.git
```

As the repository does not have an installer the directory should be added manually to the Pythonpath. This can be done from the bash profile script using a suitable text editor (Vim in the example below):

```
# Open the profile script
~$ vim ~/.profile
# Add the following line to the bottom of the profile script
export PYTHONPATH=/Users/GW/Documents/GEM/hmtk/gmpe-smtk/:$PYTHONPATH
# Save and exit
```

To complete the installation simply re-start the terminal.

2. The Ground Motion Database

2.1 Strong Motion Observations

The core structure of the GMPE-SMTK is based upon defining a reference model for describing a database of ground motion data. This means that inside of the software every database is rendered into a common, but potentially extensible, format in which the difference components of the information pertaining to a particular ground motion record is structured in a logical and readily interpretable manner. The initial structure of the "data model" is constructed to permit a widely applicable, but not exhaustive, representation of the set of attributes to describe the characteristics of the strong motion record.

2.2 Current Data Model

The comprehensive data model is shown in Figure 2.1. It's two core classes are the strong motion database (recognised internally in the toolkit as `smtk.sm_database.GroundMotionDatabase`) and the ground motion record (`smtk.sm_database.GroundMotionRecord`). The database itself only contains four attributes:

- `id`: The unique identifier of the database
- `name`: The database name
- `directory`: The location of the directory containing the database files
- `records`: The ground motion records as a list of instances of `smtk.sm_database.GroundMotionRecord`

The class also contains the method `get_contexts`. This method is fundamental to the operation of the toolkit as it translates the database into the form required for input into the OpenQuake Ground Motion Prediction Equations. To calculate the expected ground motions, using a GMPE, given the input information OpenQuake requires a set of "context" objects: `SitesContext`, `DistancesContext` and `RuptureContext`. These are a set of containers that hold information needed for OpenQuakes's GMPE's[1]. The "contexts" operate such that a single rupture can be associated with a set of sites and distances. Therefore when creating the contexts the `get_contexts` functoin will group records according to the earthquake ID and produce a set of context objects per event, not per record. **It is absolutely critical for correct usage of**

[1] Their contents will generally reflect the full list of parameters needed for all of the GMPEs currently implemented in OpenQuake. Depending on the GMPEs being considered, however, not all parameters need to be included.

the toolkit that the event IDs are correctly, and uniquely (i.e. a strong record can only be associated to one event) assigned for each record.

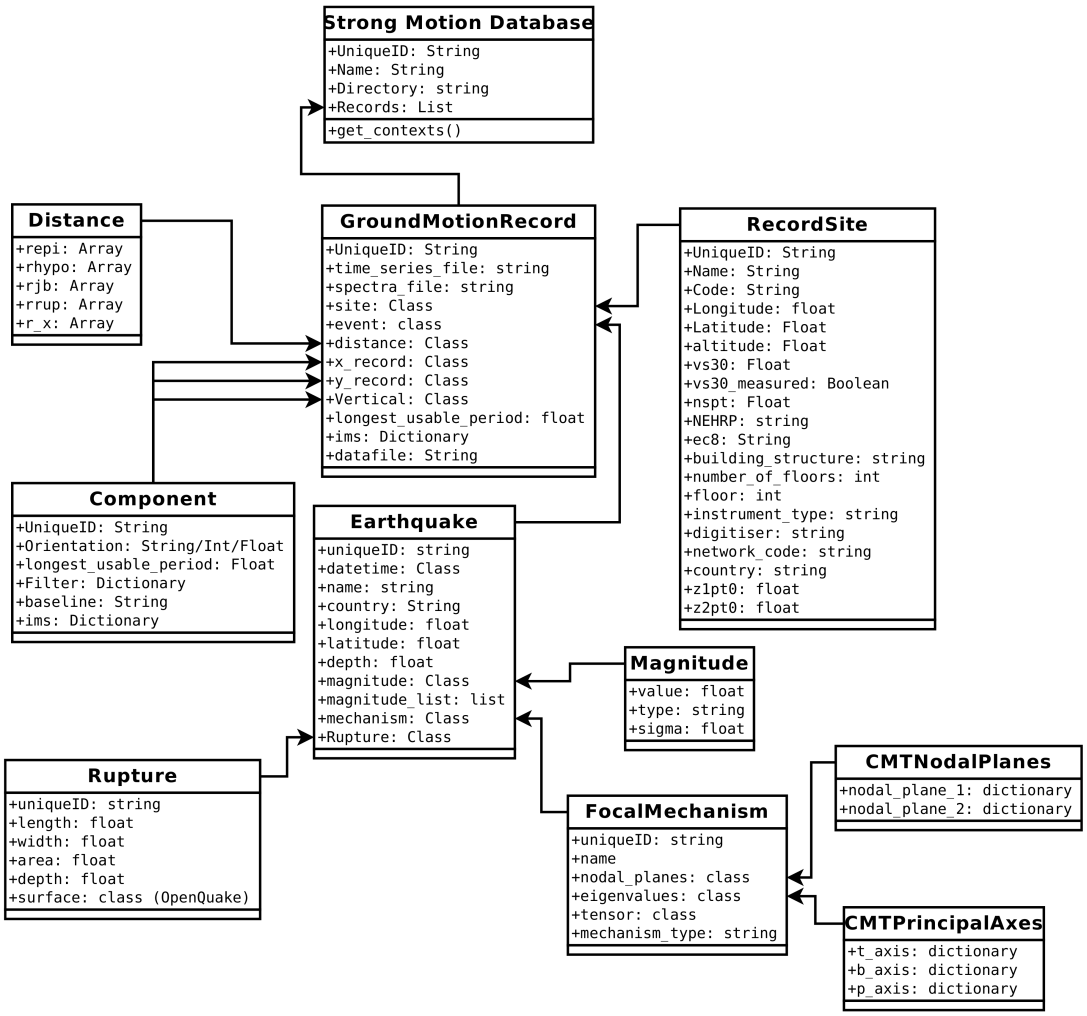


Figure 2.1 – *Core Architecture of the Strong Motion Data Model*

The `GroundMotionRecord` class is largely a single container of attributes:

- `id` Record (waveform) ID
- `time_series_file`: List of names of the files storing the time series of each of the components of the record.
- `spectra_file`: List of names of the files storing the response spectra of each of the components of the record (Optional).
- `event`: Earthquake information as an instance of the `Earthquake` class
- `distance`: Source to site distance information as an instance of the `RecordDistance` class
- `site`: Site information as an instance of the `RecordSite` class.
- `xrecord`: Metadata relating to the first horizontal component of the record as an instance of the class `Component`
- `yrecord`: Metadata relating to the second horizontal component of the record as an instance of the class `Component`
- `vertical`: Metadata relating to the vertical component of the record as an instance of the

class `Component` (Optional)
- `average_lup`: Average longest useable period of the record
- `ims`: Dictionary of scalar intensity measure attributes of the record
- `datafile`: Location of the processed hdf5 database file for the record.

2.2.1 Source Characteristics

The current characterisation of the source is broken down into several different areas: the hypocentre information, magnitude, focal mechanism and finite rupture.

Earthquake

The `Earthquake` class contains full characterisation of the event source, with the following attributes:

- `id`: Unique identifier of the event
- `datetime`: The date and time of the earthquake as an instance of Python's datetime object. This can be generated as follows:

```
from datetime import datetime
# The following event occurs at 2010/6/1 20:36:27.4
event_date_time = datetime(2010, 6, 1, 20, 36, 27, 4E5)
```

- `name`: Event Name
- `country`: Country of event (Optional)
- `longitude`: Longitude of event (in decimal degrees)
- `latitude`: Latitude of event (in decimal degrees)
- `depth`: Hypocentral depth of event (km)
- `magnitude`: Primary magnitude (usually M_W) as instance of the `Magnitude` class
- `magnitude_list`: Alternative magnitudes for the event, as list of instances of the `Magnitude` class
- `mechanism`: Earthquake focal mechanism as instance of the `FocalMechanism` class
- `rupture`: Finite rupture properties as instance of the `Rupture` class

In general the preferred magnitude scale is Moment Magnitude M_W; however to permit the possibility other scales may be preferred (or for practical purposes assumed equivalent) the `Earthquake` class contains the primary magnitude and a list of other magnitude scales. A magnitude is represented by a separate class called `Magnitude`, which contains the following attributes:

- `value` The magnitude value
- `mtype` The magnitude type
- `sigma` The standard deviation of the magnitude estimate

Where possible it is preferable to hold information relating to the focal mechanism of the class. It is anticipated that potentially several components of the focal mechanism may be needed; hence the mechanism is broken down into classes to represent the nodal planes, the eigendecomposition and the moment tensor, all held within the `FocalMechanism` class. This class has the following attributes:

- `id` Unique identifier (should be different from the event ID)
- `Name` Focal mechanism name
- `nodal_planes` The strike dip and rake of the two nodal planes of the focal mechanism as an instance of the class `GCMTNodalPlanes`. This class can be constructed as follows:

```
# Two nodal planes: i) Strike = 30., dip = 90., rake = 0.
#                   ii) Strike = 120., dip = 90., rake = 180.
from smtk.sm_database import GCMTNodalPlanes
nps = GCMTNodalPlanes
```

```
nps.nodal_plane_1 = {"strike": 30., "dip": 90., "rake" = 0.}
nps.nodal_plane_2 = {"strike": 120., "dip": 90., "rake" = 180.}
```

- `eigenvalues` The eigenvectors (B-, P- and T-) of the principal axes of the mechanism as an instance of the class `GMCTPrincipalAxes`. This can be constructed as follows:

```
# Mechanism with the following planes
# T-axis = Eigenvalue 1.5E24, Plunge = 8, Azimuth = 203
# B-axis = Eigenvalue -0.11E24, Plunge = 77, Azimuth = 332
# P-axis = Eigenvalue -1.393, Plunge = 10, Azimuth = 111
from smtk.sm_database import GCMTPrincipalAxes
pax1 = GCMTPrincipalAxes()
pax1.t_axis = {"eigenvalue": 1.5E24,
               "azimuth": 203.,
               "plunge": 8.}
pax1.b_axis = {"eigenvalue": -0.11E24,
               "azimuth": 332.,
               "plunge": 77.}
pax1.p_axis = {"eigenvalue": -1.393E24,
               "azimuth": 111.,
               "plunge": 10.}
```

- `tensor` The moment tensor as a $3\times$ array
- `mechanism_type` The style of faulting as a string

Finally the `Earthquake` class can contain information relating to the physical dimensions of the rupture (where available!). These are held in the `Rupture` class, with the following attributes:

- `id`: Unique ID (not the same as the event ID)
- `length`: Rupture length (km)
- `width`: Rupture Width (km)
- `area`: Rupture Area (km^2)
- `depth`: Depth to the top of rupture (km)
- `surface`: Rupture surface as an instance of the OpenQuake surface class

2.2.2 Site Characteristics

In contrast to the `Earthquake` class, all relevent site information is stored in just one class: `RecordSite`. This class contains the following attributes:

- `id`: Unique station identifier
- `name`: Station Name
- `code`: Station Code (as reported by station)
- `longitude`: Longitude of station
- `latitude`: Latitude of station
- `altitude`: Elevation of station (optional)
- `vs30`: V_{S30} of station
- `vs30_measured`: Boolean term to indicate if the V_{S30} is measure (True) or inferred (False) (optional)
- `vs30_measured_type`: Metadata indicating how V_{S30} is measured (optional)
- `nspt`: Number of hammer blows from standard penetration test (optional)
- `nehrp`: NEHRP site class (optional)
- `ec8`: Eurocode 8 site class (optional)
- `building_structure`: Type of building hosting the recorder (optional)
- `number_floors`: Number of floors in building hosting the recorder (optional)
- `floor`: Floor upon which the recorder is situated (optional)
- `instrument_type`: Type of recording instrument

- `digitiser`: Type of digitiser
- `network_code`: Network code
- `country`: Country of station
- `z1pt0`: Depth to V_S 1.0 km/s interface (m)
- `z2pt5`: Depth to V_S 2.5 km/s interface (km)

2.2.3 Waveform Characteristic

The database can contain information relating to the manner in which the record has been processed, in addition to other waveform-specific metadata. This informaiton is held in the `Component` class, with the following attributes:

- `id`: Identifier of the waveform
- `orientation`: Orientation of the record
- `lup`: Longest useable period
- `filter`: Information about the filter type used. For example, for a "Butterworth" filter of 2nd order, with 2 passes and a high and low-cut frequency of 100 Hz and 0.05 Hz respectively, the filter is described via the dictionary:

```
filter_data = {"Type": "Butterworth",
               "Order": 2,
               "Passes": 2,
               "Low-Cut": 0.05,
               "High-Cut": 100.0}
```

  Additional data can be added dynamically where necessary
- `baseline`: Information about baseline adjustment
- `ims`: Scalar intensity measures of the component.
- `units`: Units of the waveform

2.2.4 Geometry Characterisations

Depending on the information supplied, particularly with respect to the finite rupture, the source-to-site geometry characteristics can be stored in the `RecordDistance` class. In the present form this contains attributes of the five main distance types:

- `repi`: Epicentral distance (km) from source to site
- `rhypo`: Hypocentral distance (km)
- `rjb`: Joyner-Boore distance (km)
- `rrup`: Rupture distance (km)
- `r_x`: R_X distance (km)

2.3 Missing Metadata!

One of the greatest challenges in construction of a strong motion database is to address the cases where critical (i.e. non-optional) metadata are missing from the record.

TODO - More to add!

2.4 The HDF5 Strong Motion Database

The potentially widespread application of the software requires that the manner in which it is able to create and access waveform data is important to the practical function of the tools. Therefore the decision is made to store the waveforms and related attributes in an `hdf5` database [2]. `hdf5` is a high density binary file format capable of storing data in a nested architecture,

[2] http://www.hdfgroup.org/HDF5/

allowing large arrays of data (such as waveforms) to be mapped alongside their attributed This database consists of two components. The metadata for each record, mapped according to the `GroundMotionDatabase` class, and the waveform data itself. The waveform data for a database is split into a directory of `hdf5` binary files, where each file contains the full waveforms for all three components of the record, as well as the spectra for each waveform, and potentially the horizontally resolved spectra. The full structure of a single ground motion record in a binary format is shown in Figure 2.2.

The `hdf5` format offers several key advantages that have motivated its usage within these tools. Already mentioned was the capability to store large arrays of numerical data in a structure that is conveniently nested in order to associate subsets of data and corresponding attributes. The format is high-density, reducing the overall size of the datafile compared to, for example, plain text ascii format. The file structure is also dynamic allowing for attributes and dataset to be added as needed. This enables the user to undertake the data compilation in several steps, adding on processed data subsequently from the initial compilation stage. The `hdf5` format is widely supported by many different software languages, and its contents visualisable via the cross-platform `HDFView` application [3].

2.4.1 Loading Data to the Database

In general the creation of a database from a set of records is a process that cannot be readily generalised as each strong motion recording network will define its own data format. Whilst progress is ongoing to add database creators for some of the largest strong motion recording networks, it is necessary for the user to create a customised reader for their own networks. The readers constitute a set of three separate modules, the outline of which can be found in the `smtk.parsers.base_database_parser` module: i) `SMDatabaseReader` the main reader to build the metadata structure of the database (i.e. the `GroundMotionDatabase` class), ii) `SMTimeSeriesReader` to read the waveforms from the respective time series files, and iii) `SMSpectraReader` to read the spectra from files (if provided by the network). If the corresponding database creators are available then a full database can be created using the module `smtk.sm_database_builder.SMDatabaseBuilder`, using the example below:

```
# Import the readers for database XXXXX
from smtk.parsers.XXX_database_parser import (XXXMetadataReader,
                                              XXXTimeSeriesReader,
                                              XXXSpectraReader)
# Import the database builder
from smtk.sm_database_builder import SMDatabaseBuilder,

# Path to the directory containing the input files
input_fileset = "./path/to/strong/motion/folder"
# Path to the directory intended for the database
output_database = "./path/to/output/database/folder"

# Create an instance of the database builder
db_builder = SMDatabaseBuilder(XXXMetadataReader,
                               output_database)

# Create the metadata database
db_builder.build_database("UNIQUEID",  # Database ID
                          "DATABASE NAME",  # Name
                          input_fileset)

```

[3] http://www.hdfgroup.org/products/java/hdfview/

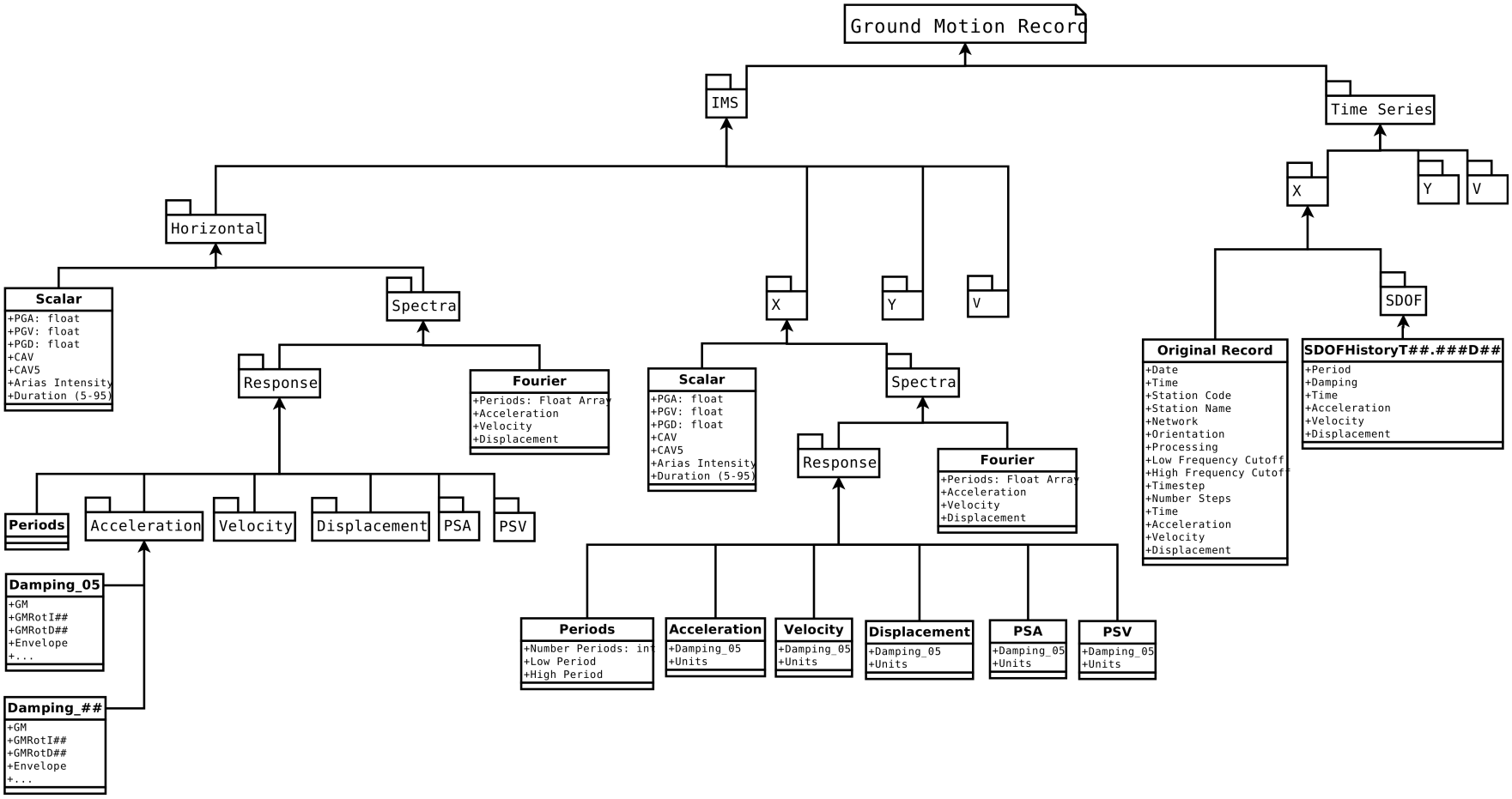


Figure 2.2 – *Architecture of the hdf5 representation of a strong motion record*

```
# Parse the strong motion records
db_builder.parse_records(XXXTimeSeriesReader,
                         XXXSpectraReader)

# (This may take some time if there are many records!!!)
```

The time taken to create the database will obviously depend on many factors, including the number of records and the sampling frequency of each record. For large databases however it could potentially take many hours!

When compiled the database will store the metadata in a file inside the database directory named "`metadata.pkl`". This file is a Python binary file (called a "Pickle" file or "pkl") that stores the `GroundMotionDatabase` class. To load this data into the workspace simply do the following:

```
# Import the cPickle module
import cPickle
db1 = cPickle.load(open("path/to/database/metafile.pkl", "r"))
```

2.5 Selecting Subsets of Records from the Database

For many of the GMPE tools that will be shown in the rest of this book there will be a frequent need to consider only subsets of the full database. To do this a set of selection tools are created to facilitate the database selection process. These tools can be found in the module `smtk.strong_motion_selector.SMRecordSelector`. To select a subset of the database it is simply necessary to do the following:

```
# Import the selector
from smtk.strong_motion_selector import SMRecordSelector
# Create an instance of the class for a specific database
selector = SMRecordSelector(db1)

# Select the 3rd, 6th and 17th record in the database
db2 = selector.select_records([2, 5, 16], as_db=True)
```

The keyword `as_db` tells the function whether to return the selected records as a new instance of the `GroundMotionDatabase` class. In most cases it is advisable to set this to true, otherwise it will return the selected records as a list of `GroundMotionRecord` classes.

The `SMRecordSelector` has the following selection methods

- `select_from_record_id(self, record_id)`

  Selects a record according to its waveform ID.
- `select_from_event_id(self, event_id, as_db=False)`

  Returns a set of records from a common event defined by its `event_id`.
- `select_within_time(self, start_time=None, end_time=None, as_db=False)`

  Selects records within a specific time window, defined as the interval between `start_time` and `end_time` (both Python datetime objects). For example:

```
# Select records between 1990/01/01 and 2010/12/31
db2 = selector.select_within_time(datetime(1990, 1, 1),
                                  datetime(2010, 12, 31),
                                  as_db=True)
```

- `select_within_depths(self, upper_depth=None, lower_depth=None, as_db=False)`

  Selects records corresponding to events within a specific depth range defined as the interval between `upper_depth` (km) and `lower_depth` (km). If not specified `upper_depth` defaults to zero, and lower depth defaults to ∞. Records missing a hypocentral depth will not be selected, therefore setting both limits to `None` will effectively filter these records out of the database.
- `select_within_magnitude(self, lower=None, upper=None, as_db=False)`

  Select records corresponding to events within a magnitude range, defined as the interval between `lower` (defaults to $-\infty$ if not specified) and `upper` (defaults to ∞ if not specified).
- `select_by_station_country(self, country, as_db=False)`

  Returns the records within a specific country (input as a string). For example, to select only observations recorded in Italy:

```
db2 = selector.select_by_station_country("Italy",
                                         as_db=True)
```

- `.select_by_site_attribute(self, attribute, value, as_db=False)`

  Select records corresponding to a particular site attribute (according the attributes listed in the `RecordSite` class). For example to select records only recorded at "Free-Field" stations:

```
db2 = selector.select_by_site_attribute("building_structure",
                                        "Free-Field",
                                        as_db=True)
```

- `select_within_vs30_range(self, lower_vs30, upper_vs30, as_db=False)`

  Select records within a given Vs30 range defined as the interval between `lower_vs30` and `upper_vs30`, which default to $-\infty$ and ∞ if not specified. As with the depth selection, setting both `lower_vs30` and `upper_vs30` to `None` will remove records missing V_{S30} values from the database.
- `select_stations_within_distance(self, location, distance, as_db=False)`

  Selects records from stations within a distance (km) of a specified location. The location must be input as an instance of an OpenQuake "Point" class. For example, to select records within 200 km of 30°E and 40°N:

```
from openquake.hazardlib.geo.point import Point
location = Point(30.0, 40.0)
db2 = selector.select_stations_within_distance(location,
                                               200.0,
                                               as_db=True)
```

- `select_stations_within_region(self, region, as_db=False)`

  Selects station inside of a specified region defined by a polygon. The polygon must be input as an instance of the OpenQuake Polygon class. For example, to select all ground motions recorded within a square bounded by 30°E to 40°E and 20°N to 40°N:

```
from openquake.hazardlib.geo.point import Point
```

```
from openquake.hazardlib.geo.polygon import Polygon

region = Polygon([Point(30.0, 40.0),
                  Point(40.0, 40.0),
                  Point(40.0, 20.0),
                  Point(30.0, 20.0)])
db2 = selector.select_stations_within_region(region,
                                             as_db=True)
```

- `select_within_distance_range(self, distance_type, shortest, furthest, alternative=False, as_db=False)`
  Select records based on a source to site distance range, with the distance type specified by `distance_type`, within the interval `shortest` (defaults to 0 if not specified) and `furthest` (defaults to ∞ if not specified). Recognising the possibility that some distance metric may be missing it is possible to also specify a second metric and corresponding distance limits as a tuple via the `alternative` keyword. For example, select records within a Joyner-Boore distance of 5 km and 50 km, and, if Joyner-Boore distance is not specified select records within an epicentral distance of 10 km and 70 km.

```
db2 = selector.select_within_distance_range(
    "rjb", 5.0, 50.0,
    alternative=("repi", 10.0, 70.0),
    as_db=True)
```

- `select_mechanism_type(self, mechanism_type, as_db=False)`
  Select records based on a descriptive event mechanism type. For example, to select records from "strike-slip" events:

```
db2 = selector.select_mechanism_type("strike-slip",
                                     as_db=True)
```

- `select_event_country(self, country, as_db=False)`

  Select records corresponding to events whose epicentres occur within a specific country
- `select_epicentre_within_distance_from_point(self, location, distance, as_db=False)`

  Selects records from earthquakes whose epicentres are within a distance of a point. The point must be input as an instance of the OpenQuake "Point" class.
- `select_epicentre_within_region(self, region, as_db=False)`

  Selects records from event inside the specified region. The region must be defined as an instance of the OpenQuake "Polygon" class.
- `select_longest_usable_period(self, lup, as_db=False)`

  Selects records with a longest usable period greater than or equal to `lup` (in seconds)

3. Response Spectra and Intensity Measures

3.1 Ground Motion Waveforms

Fundamental to the use of the GMPE-SMTK is the strong motion waveform. For the widest portability we adopt the simplest representation of the waveform in the toolkit, which requires only the acceleration and the time steps. As seen in the previous chapter, the strong motion database stores the acceleration, velocity and displacement trace.

The conventional units for acceleration, velocity and displacement waveforms in the GMPE-SMTK are "cm/s/s", "cm/s" and "cm" respectively.

The GMPE-SMTK contains two tools for extracting basic information and common ground motion intensity measures from single waveforms, or from a horizontal pair of waveforms: `intensity_measures` and `response_spectrum`. To use them in an application we can import them as follows:

```
import smtk.intensity_measures as ims
import smtk.response_spectrum as rsp
```

This will load two objects into the workspace `ims` (the intensity measure tools) and `rsp` (the response spectrum tools)

In most applications we assume it is the acceleration time series that is available. In the following example we consider a pair of strong motion records, where each record is represented by a simple ascii (text) file:`sm_record_x.txt` and `sm_record_y.txt`. The timestep of each record is 0.002 s.

```
# Import the numpy tools
import numpy as np
# Load in the records
accel_x = np.genfromtxt("sm_record_x.txt")
accel_y = np.genfromtxt("sm_record_y.txt")
time_step = 0.002
```

Once loaded the user can retrieve the velocity and displacement time-series, calculated using double integration of the acceleration time series. A simple tool to do this is found in the `ims` tools:

```
vel_x, disp_x = ims.get_velocity_displacement(accel_x,
                                              time_step,
                                              units="cm/s/s")
```

where `units` is the units of the uploaded record.
The first steps one might wish to take when analysing a strong motion record is to look at the waveforms. To do this we make use of the simple tool inside the response spectrum package called `plot_time_series`. This can be called as follows for the x=component of the pair:

```
rsp.plot_time_series(accel_x,
                     time_step,
                     velocity=vel_x,
                     displacement=disp_x,
                     units="cm/s/s",
                     filename="path/to/output/image.eps",
                     filetype="eps",
                     dpi=300,
                     linewidth=1.5)
```

This command would produce a plot of the style shown in Figure 3.1.

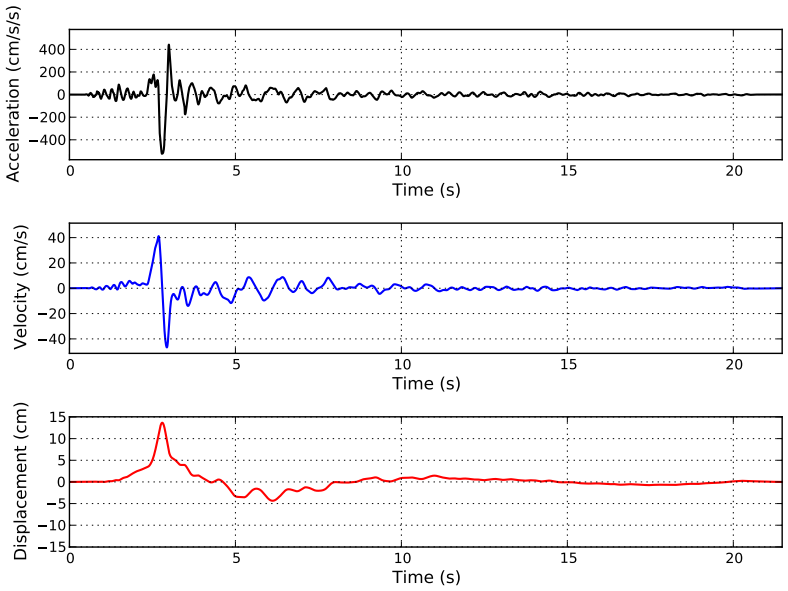


Figure 3.1 – *Example time series for a single record: acceleration (top), velocity (middle) and displacement (bottom)*

The function `plot_time_series` requires two essential arguments:

- `acceleration`: The acceleration time series
- `time_step`: The time step of the acceleration time series

If the user already has the velocity and displacement time series available these can be input with the following keyword arguments:

- `velocity`: The velocity time series (cm/s), which will be calculated if not provided.
- `displacement`: The time step of the displacement (cm) time series, which will be calculated if not provided.
- `units`: The units of the acceleration time series (defaults to "cm/s/s")
- `figure_size`: The internal size of the figure (defaults to $(8,6)$)
- `filename`: The name of the file to which the figure will be saved (if desired, by default it is "None")
- `filetype`: The format of the output file (default="png").
- `dpi`: Resolution (dots per inch) of the output figure (default = 300)
- `linewidth`: The thickness (point size) of the lines (default = 1.5)

3.2 Scalar Intensity Measures

3.2.1 Peak Measures

The following describes the simple "peak" measures, peak ground acceleration (PGA), peak ground velocity (PGV) and peak ground displacement (PGD), that can be extracted from the record:

$$\begin{aligned} PGA &= \max|a(t)| \\ PGV &= \max|v(t)| \\ PGD &= \max|d(t)| \end{aligned} \tag{3.1}$$

where $a(t)$, $v(t)$ and $d(t)$ are the acceleration, velocity and displacement time series respectively.

These values can all be extracted by one function in the `ims` tool named `get_peak_measures`. These values can be called as follows:

```
pga, pgv, pgd, vel_x, disp_x = get_peak_measures(time_step,
                                                 accel_x,
                                                 get_vel=True,
                                                 get_disp=True)
```

This function returns the three peak measures as well as the velocity and displacement (if required). The keywords `get_vel` and `get_disp` will, when set to `True`, calculate velocity and displacement.

3.2.2 Duration

3.2.3 Other Scalar Measures

3.3 Response Spectra

The response spectrum is one of the most important elements of a ground motion record used in earthquake engineering. The response spectrum provides the peak responses of a set a single-degree-of-freedom (SDOF) oscillators with different natural periods (T), and damping ratio ξ. The pseudospectral acceleration at a given period $Sa(T,\xi)$, and its corresponding pseudospectral velocity and displacements, $Sv(T,\xi)$ and $Sd(T,\xi)$ respectively, are the most widely used means to characterise the ground motion input for a given structure. In addition to the peak values however, it is also useful to have available the acceleration, velocity and displacement response time series of the SDOF oscillators for each natural period. The GMPE-SMTK response spectra tools are designed to return the most comprehensive set of information to describe the SDOF response for each record.

To calculate a response spectrum for a record, two response spectra calculators are currently available: "Newmark-β" and "Nigam & Jennings" (CITE NigamJennings1969).

These can be called following the process below:

```
# Define the periods for calculating spectral acceleration
periods = np.array([0.01, 0.02, 0.03, 0.04, 0.05, 0.075, 0.1,
                    0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17,
                    0.18, 0.19, 0.20, 0.22, 0.24, 0.26, 0.28,
                    0.30, 0.32, 0.34, 0.36, 0.38, 0.40, 0.42,
                    0.44, 0.46, 0.48, 0.50, 0.55, 0.6-, 0.65,
                    0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00,
                    1.10, 1.20, 1.30, 1.40, 1.50, 1.60, 1.70,
                    1.80, 1.90, 2.00, 2.20, 2.40, 2.60, 2.80,
                    3.00, 3.20, 3.40, 3.60, 3.80, 4.00, 4.20,
                    4.40, 4.60, 4.80, 5.00, 5.50, 6.00, 6.50,
                    7.00, 7.50, 8.00, 8.50, 9.00, 9.50, 10.0])
```

```
# Call the Newmark-Beta methodology
newmark_beta = rsp.NewmarkBeta(acceleration,
                               time_step,
                               periods,
                               damping=0.05,
                               units="cm/s/s")

spectra, time_series, acc, vel, dis = newmark_beta.evaluate()

# Call the Nigam & Jennings (1969) method
nigam_jennings = rsp.NigamJennings(acceleration,
                                   time_step,
                                   periods,
                                   damping=0.05,
                                   units="cm/s/s")

spectra, time_series, acc, vel, dis = nigam_jennings.evaluate()
```

Each response spectrum method requires the following input:

- `acceleration`: The acceleration time-series
- `time_step`: The time step (s)
- `periods`: An array of natural periods
- `damping`: The fractional damping ratio of the oscillator (defaults to 0.05 if not specified)
- `units`: The units of the acceleration time-series (defaults to "cm/s/s")

The calculators produce five outputs:

1. `spectra`: A dictionary with the following keys:
   - `Period`: The vector or spectral periods
   - `Acceleration`: The peak acceleration response at each period (cm/s/s)
   - `Velocity`: The peak velocity response at each period (cm/s)
   - `Displacement`: The peak displacement response at each period (cm)
   - `Pseudo-Acceleration`: The peak pseudo-acceleration response at each period (cm/s/s), where pseudo-acceleration is defined as:

$$PSa(T,\xi) = \frac{4\pi^2}{T^2} Sd(T,\xi) \tag{3.2}$$

   - `Pseudo-Velocity`: The peak pseudo-velocity response at each period (cm/s/s), where pseudo-velocity is defined as:

$$PSv(T,\xi) = \frac{2\pi}{T} Sd(T,\xi) \tag{3.3}$$

2. `time_series`: A dictionary with the following items:
   - `Time-Step`: The time step of the record
   - `Acceleration`: The acceleration time-series (cm/s/s) of the original record
   - `Velocity`: The velocity time-series (cm/s) of the original record
   - `Displacement`: The displacement time-series (cm) of the original record
   - `PGA`: The peak ground acceleration of the record (cm/s/s)
   - `PGV`: The peak ground velocity of the record (cm/s)
   - `PGD`: The peak ground displacement of the record (cm)
3. `acc`: A 2D array in which each column contains the acceleration time-series of the SDOF oscillator response to the record for each period.
4. `vel`: A 2D array in which each column contains the velocity time-series of the SDOF oscillator response to the record for each period.

5. `dis`: A 2D array in which each column contains the displacement time-series of the SDOF oscillator response to the record for each period.

If, alternatively, one does not wish to import the response spectra methods manually then the `ims` tools have a function to apply this process. The same results as those shown previously can be obtained from this tool as follows:

```
spectra, time_series, acc, vel, dis = ims.get_response_spectrum(
    accel_x,
    time_step,
    periods,
    damping=0.05,
    units="cm/s/s",
    method="Nigam-Jennings")
```

The inputs are the same as for the response spectrum calculators except for `method`, which indicates the preferred method for calculating the response spectrum. At present this is either `Newmark-Beta` or `Nigam-Jennings`, which the latter selected as the default.

To view the full set of resulting response spectra, the `rsp` tool has a method names `plot_response_spectra`, which is used as follows and will produce a plot similar to that of Figures 3.2 and 3.3:

```
rsp.plot_response_spectra(spectra,
                          axis_type="loglog")
```

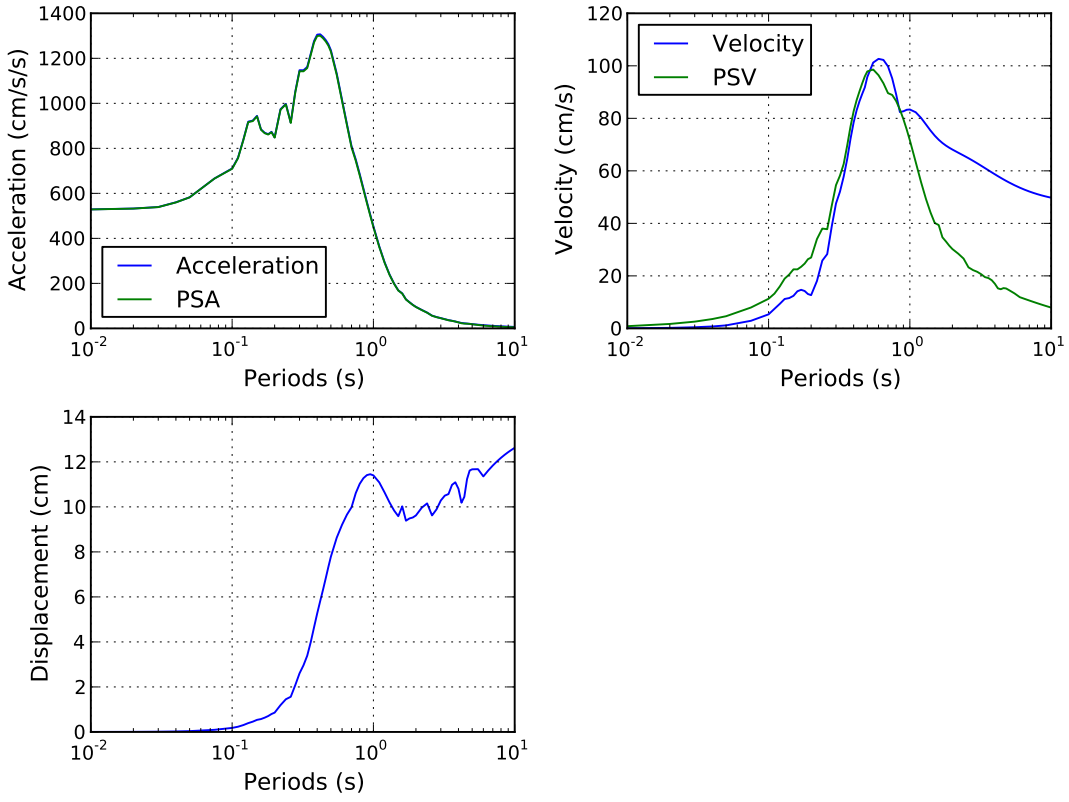


Figure 3.2 – *Full response spectra of the waveform shown in Figure 3.1, calculated using the Newmark-β method*

where `spectra` is the response spectra dictionary, `axis_type` defines the type of axes: “loglog” (double logarithmic - the default), “semilogx” (logarithmic period axis), “semilogy” (logarithmic

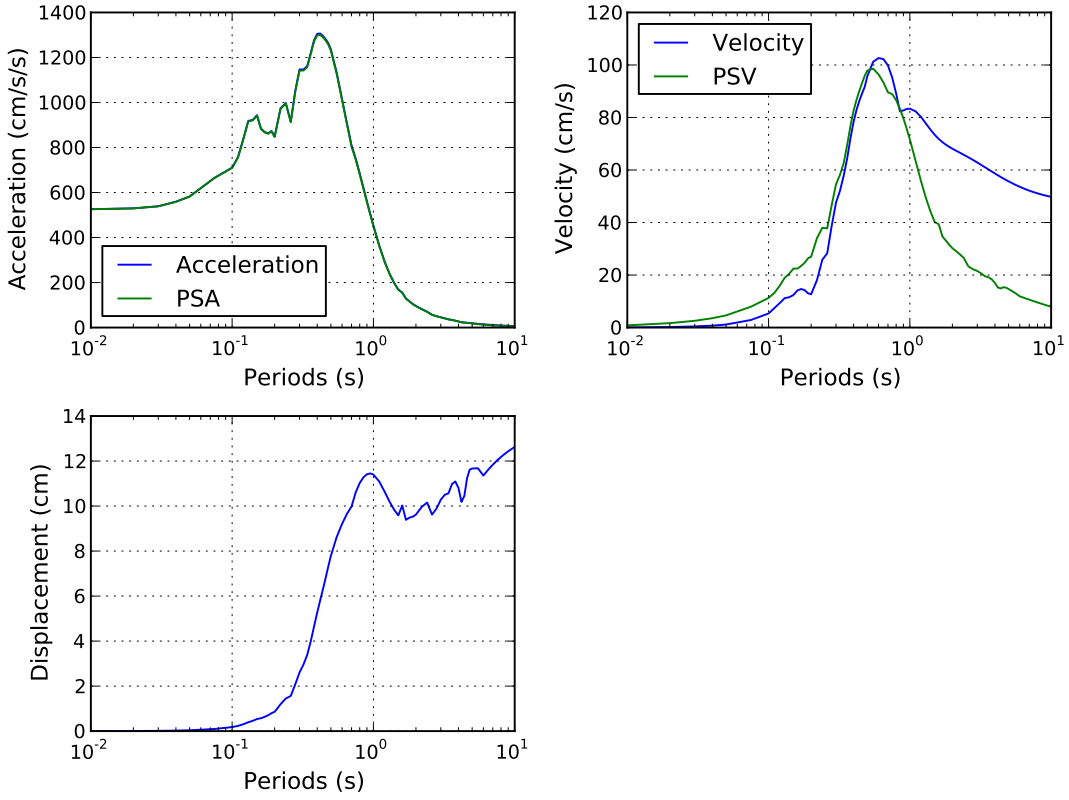


Figure 3.3 – *Full response spectra of the waveform shown in Figure 3.1, calculated using the Nigam & Jennings (1969) method*

spectral response axis), "linear" (both axes linear). The function also takes the same keywords as the `plot_time_series` method, i.e. `figure_size`, `filename`, `filetype`, `dpi`.

It can be seen from Figures 3.2 and 3.3 that both methods give very similar results.

3.4 Combining Horizontal Components of Motion

For a pair of horizontal records it is necessary to define a spectrum representing the "horizontal" component of ground motion. The manner in which the two horizontal components of motion are resolved can be treated in many different ways (cite: Douglas, 2003, ????; Beyer and Bommer etc.). The intensity measure tools contain various methods for combining horizontal response spectra.

The first step in the application of these methods may be to extract the respective response spectra for the two horizontal components. This can be done as follows:

```
spectra_x, spectra_y = ims.get_response_spectrum_pair(
    accel_x,
    time_step, # Time-step of x-component
    accel_y,
    time_step, # Time-step of y-component,
    periods,
    damping=0.05,
    units="cm/s/s",
    method="Nigam-Jennings")
```

In the case of the records demonstrated the time-step is the same for both horizontal components. If it is not the same, however, the response spectra can still be calculated.

Given the pair of records, `spectra_x` ($Sa_x(T,\xi)$)and `spectra_y` ($Sa_y(T,\xi)$), we can now obtain the following "resolved" horizontal spectra:

- **Geometric Mean**

$$Sa_{gm}(T,\xi) = \sqrt{Sa_x(T,\xi) \times Sa_y(T,\xi)} \tag{3.4}$$

Calculated using the following command:

```
sa_gm = ims.geometric_mean_spectrum(spectra_x, spectra_y)
```

- **Arithmetic Mean**

$$Sa_{am}(T,\xi) = \frac{1}{2}(Sa_x(T,\xi) + Sa_y(T,\xi)) \tag{3.5}$$

Calculated using the following command:

```
sa_am = ims.arithmetic_mean_spectrum(spectra_x, spectra_y)
```

- **Larger PGA** This simply returns the spectrum of the time series that gives the largest spectral acceleration. Calculated using the following command:

```
sa_larger = ims.larger_pga(spectra_x, spectra_y)
```

- **Envelope** This returns a spectrum representing the larger of the two components for each period such that:

$$Sa_{env}(T_i,\xi) = \max(Sa_x(T_i,\xi), Sa_y(T_i,\xi)) \quad \text{for} \quad i = 1,2,\ldots,N_{PERIODS} \tag{3.6}$$

```
sa_env = ims.envelope(spectra_x, spectra_y)
```

In each of these cases the output of the function is a dictionary containing the same keys as that of the `spectra`, but with the resolved horizontal values of each of the quantities.

For the two spectra considered here, the geometric mean spectra and the envelope spectra are shown in Figures 3.4 and 3.5 respectively.

3.5 Rotation of the Horizontal Records

In some applications it may be necessary to rotate a pair of horizontal time series, either to orient them into fault-normal and fault-parallel components, or to some direction that may represent the most adverse for a particular structure. For two time series of equal duration and time series, the records can be rotated through angle θ:

$$\begin{pmatrix} a_{x(\theta)}(t) \\ a_{y(\theta)}(t) \end{pmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} a_x(t) \\ a_y(t) \end{pmatrix} \tag{3.7}$$

This can be achieved using the `ims` tools:

```
rot_hist_x, rot_hist_y = ims.rotate_horizontal(accel_x,
                                               accel_y,
                                               theta)
```

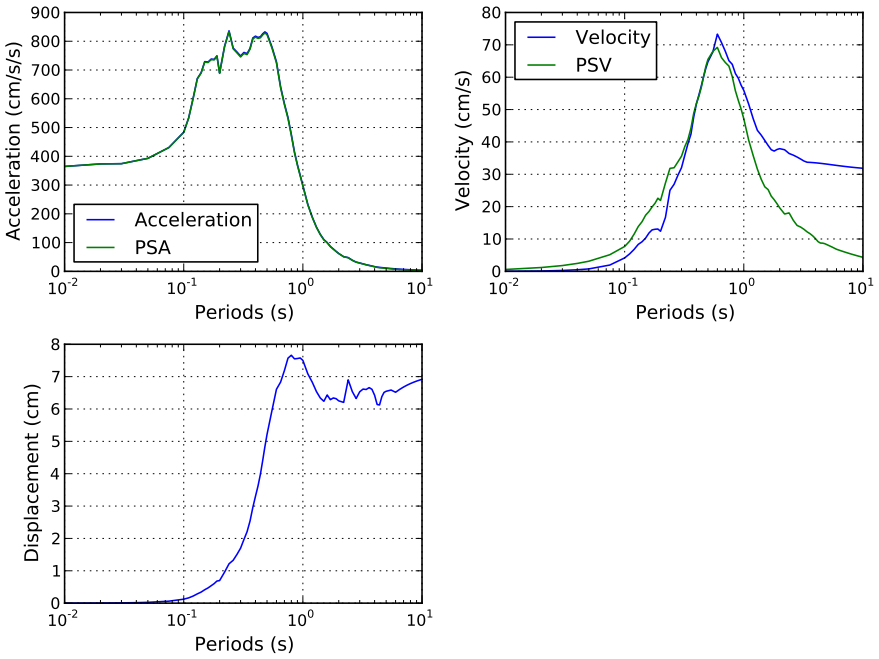


Figure 3.4 – *Geometric mean spectra of the two horizontal records*

The incorporation of rotational tools permits the toolkit to calculate the "orientation-dependent" and "orientation-independent" geometric mean of a horizontal pair of records, as described in detail in CITE Boore et al (2006). The two quantities, `GMRotDpp` and `GMRotIpp` respectively, cannot be calculated from the two horizontal response spectra directly as other horizontal measures are. As they require rotation of time-series through non redundant angles, it is necessary to determine the response spectra for each rotation angle.

To calculate `GMRotDpp` and `GMRotIpp` run:

```
gmrotd50 = ims.gmrotdpp(accel_x, time_step,
                        accel_y, time_step,
                        periods,
                        50.0, # Percentile
                        damping=0.05,
                        units="cm/s/s",
                        method="Nigam-Jennings")
gmroti50 = ims.gmrotipp(accel_x, time_step,
                        accel_y, time_step,
                        periods,
                        50.0, # Percentile
                        damping=0.05,
                        units="cm/s/s",
                        method="Nigam-Jennings")
```

3.6 Adding Horizontal Components to the Database

In the process of constructing the database of strong motion records it may often be prudent to compile the database in steps, of which the final one may often be the addition of the resolved horizontal components of the strong motion record. Depending on the data, and the required intensity measures, the calculation of horizontal spectra may be a (relatively) slow process with respect to the rest of the database construction. It is recommended that the database be

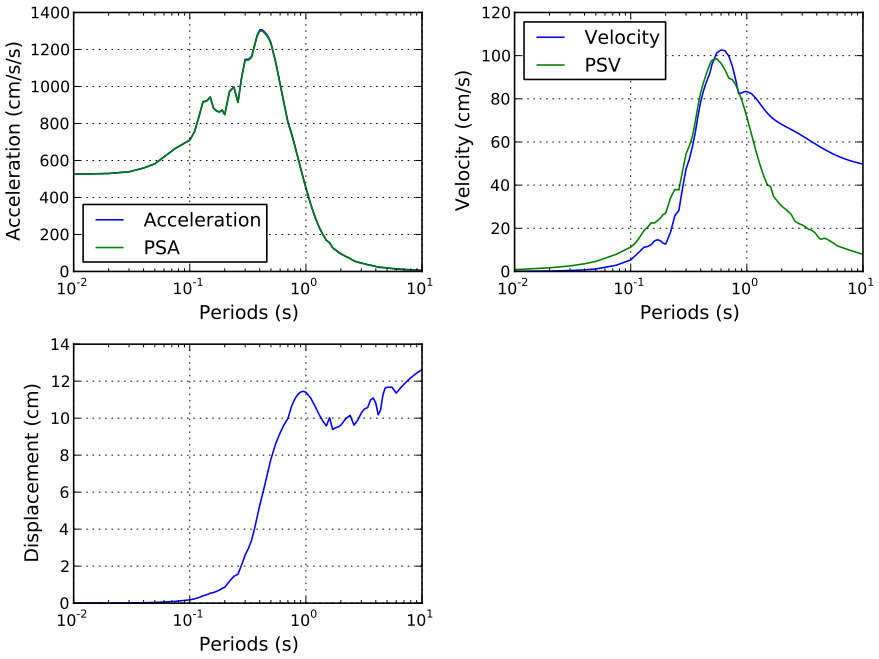


Figure 3.5 – *Envelope spectra of the two horizontal records*

constructed in the manner described in section 2.4, and then the resolved horizontal components added subsequently. To facilitate this a function is added to the database builder, which will add the desired horizontal ground motions to an existing database. An example is seen below:

```
# Import the function add_horizontal_im
from smtk.sm_database_buider import add_horizontal_im
# Load in the database metadata file
import cPickle
database = cPickle.load(open("path/to/metadata.pkl", "r"))
# Choose the desired horizontal intensity measures
# The following will add the horizontal PGA, PGV
# Geometric mean spectrum, envelope spectrum and GMRotI50
im_list = ["PGA", "PGV", "Geometric", "Envelope", "GMRotI50"]
add_horizontal_im(database,
                  im_list,
                  component="Geometric",
                  damping="05",
                  periods=[])
```

Depending on the size of the database this may take **many hours** to run (especially if rotational parameters are required). In this function the keyword `component` refers only to the horizontal component of the scalar values. For the spectra, the type of horizontal component should be given in the intensity measure list. Note also that unlike the `ims` and `rsp` tools in which the damping is input as a floating point value, here it is input as a string. If not input by the user, the periods will be taken from the values stored in the x-component of the record.

4. Comparing GMPEs: Trellis Plotting

4.1 Trellis Plots: Advantages & Limitations

The selection of GMPEs for use in probabilistic seismic hazard analysis should require an understanding of the manner in which each GMPE characterises the ground motion scaling with respect to the properties of the seismic source, and the attenuation of the motion with respect to distance. This information can help in the interpretation of the seismic hazard results, particularly those related to disaggregation, in order to best understand how the ground motion prediction equation can influence the seismic hazard at a site.

Trellis plots can be a useful tool in this process as they permit the hazard modeller to compare multiple GMPEs under a variety of conditions and understand how the GMPEs differ in terms of their fundamental source, path and site characteristics. The GMPE-SMTK provides a set of tools to provide the modeller the ability to compare OpenQuakes GMPE implementations under a variety of conditions.

One of the challenges in developing trellis plots is to compare, in a quantitatively meaningful sense, GMPEs that require different characteristics of the source, path and site. One of the most common areas of divergence can be in terms of the metric use to measure source to site distance. Many modern GMPEs for active shallow crustal regions adopt Joyner-Boore distance (shortest distance to the surface projection of the rupture) as the predictor variable for describing attenuation with distance, whilst others adopt rupture distance (the shortest distance from the site to the rupture surface), and several now adopt more complex forms that require characterisation of several types of source to site metric in order to capture the effects of directivity and hanging-wall scaling of the ground motion. This can be complicated further when we consider that the relation between source and distances metrics will vary depending on the scaling of the rupture with magnitude and the geometric orientation of the rupture. Typical trellis plots, whilst commonly used in studies of probabilistic seismic hazard, rarely seem to explicitly consider the differences in the relation between different distance metrics, except via the use of empirical relations, and their dependence on the physical characteristics of the source and site configuration.

The GMPE-SMTK attempts to address this issue by providing two contexts in which to compare GMPEs. The first is a "non-specific rupture" context in which the physical dimensions of the rupture are not explicitly considered. In this case, the user must specify the precise distances *a priori*, thus requiring them to determine the geometrical relationship between the metrics by other means. The second type is a "specific rupture" context, in which the user

provides sufficient information in order to define a planar rupture with dimensions consistent with the magnitude of the earthquake. From this rupture the GMPE-SMTK uses OpenQuake's own geometry functions to calculate all the required source-to-site distances with respect to the rupture plane. This ensures that not only is the implicit relation between different attenuation metrics accurately defined, it also guarantees that the calculation is done consistently between the trellis plotting tools and the PSHA software that may be ultimately used to implement the GMPEs being compared.

The trellis plotting tools are all contained in the function `smtk.trellis.trellis_plots`, which we shall import as follows:

```
import smtk.trellis.trellis_plots as trpl
```

and which we will refer to as `trpl` hereafter.

If the Openquake-hazard library is installed in your computer, the full list of available GMPEs can be retrieved in the following manner:

```
# Import the get_available_gsims function from OpenQuake
from openquake.hazardlib.gsim import get_available_gsims
# Show list of gsims
get_available_gsims()
```

In the examples shown throughout this chapter we consider six GMPES: citeAkkarBommer2010, citeAkkarCagnan2010, site Akkar2014 (Joyner-Boore coefficients), citeBooreAtkinson2008, citeChiouYoungs2008, citeZhaoEtAl2006. We also compare using just four intensity measures: PGA, $Sa\,(0.2s)$, $Sa\,(1.0s)$ and $Sa\,(2.0s)$.

4.2 Trellis Plotting: The (Not So) Simple Way

The non-specific rupture context requires no additional tools besides the `trpl` function. However, it is necessary that the user specifies, manually, the distance metrics, and other rupture information that may be absent.

This could be done as follows:

```
gmpe_list = ["AkkarBommer2010",
             "AkkarCagnan2010",
             "AkkarEtAlRjb2014",
             "BooreAtkinson2008",
             "ChiouYoungs2008",
             "ZhaoEtAl2006Asc"]

imts = ["PGA", "SA(0.2)", "SA(1.0)", "SA(2.0)"]
params = {"ztor": 5.0,    # Top of rupture depth
          "hypo_depth": 10.0,    # Hypocentral depth
          "vs30": 800.0, # Vs30 for all sites
          "vs30measured": True, # Vs30 value is measured
          "z1pt0": 100.0, # Depth (m) to the 1.0 km/s Vs interface
          "dip": 90.0,  # Vertical Fault
          "rake": 0.0 # Strike-slip fault
          }
```

The GMPEs and intensity measures must be input as a list, whilst missing parameters a grouped together in a Python dictionary.

4.2.1 Comparing the Scaling with Magnitude

The first trellis tool simply shows how the ground motion intensity value scales with respect to the magnitude of the rupture. For the above GMPEs we consider magnitudes from M_W 4.5 to

M_W 8.0, every 0.1 magnitude units. We also assume that the ground motions correspond to a site located at a Joyner-Boore distance of 15 km away from the vertical dipping fault (with a top of rupture depth of 5 km). We assume that the epicentral distance is equal to 2 km. This configuration is specified like so:

```
# Import the numerical python tool
import numpy as np
# Generate magnitudes from 4.5 to 8.0 every 0.1
magnitudes = np.arange(4.5, 8.1, 0.1)
distances = {"repi": 20.0,
             "rhypo": 22.5,
             "rjb": 15.0,
             "rrup": 16.0,
             "rx": 15.0}
```

Distances must be specified in a Python dictionary containing one or more of the following keys: `repi` (Epicentral Distance), `rhypo` (hypocentral distance), `rjb` (Joyner-Boore Distance), `rrup` (Rupture distance) and `rx` (Rx distance). Whilst not all distance metrics may be needed the tools with check the selected GMPEs and the distances dictionary provided. If any distance metric required by any of the GMPEs is not found in the distances input then an error will be raised.

The trellis plots can be generated using the following command, which will produce the plot shown in Figure 4.1

```
trpl.MagnitudeIMTTrellis(magnitudes,
                         distances,
                         gmpe_list,
                         imts,
                         params,
                         figure_size=(7,5),
                         filename="path/to/plot",
                         filetype="png")
```

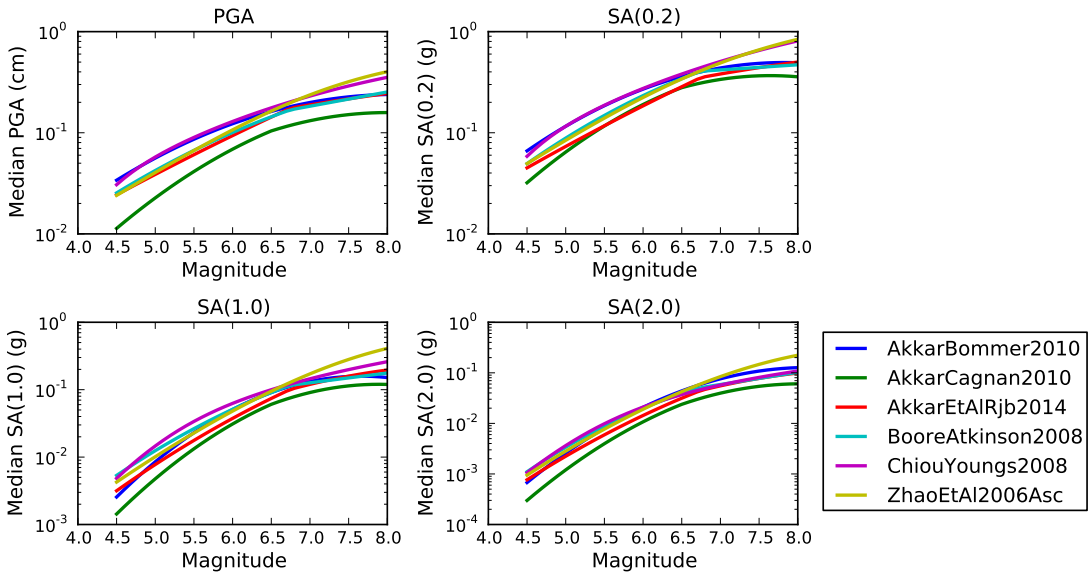


Figure 4.1 – *Scaling of GMPEs with respect to magnitude*

In addition to viewing the scaling of the expected ground motion from the GMPE, it is also possible to view the scaling of the standard deviation. This can be done with the following

command:

```
trpl.MagnitudeSigmaIMTTrellis(magnitudes,
                              distances,
                              gmpe_list,
                              imts,
                              params,
                              stddevs="Total",
                              figure_size=(7,5),
                              filename="/path/to/plot",
                              filetype="png")
```

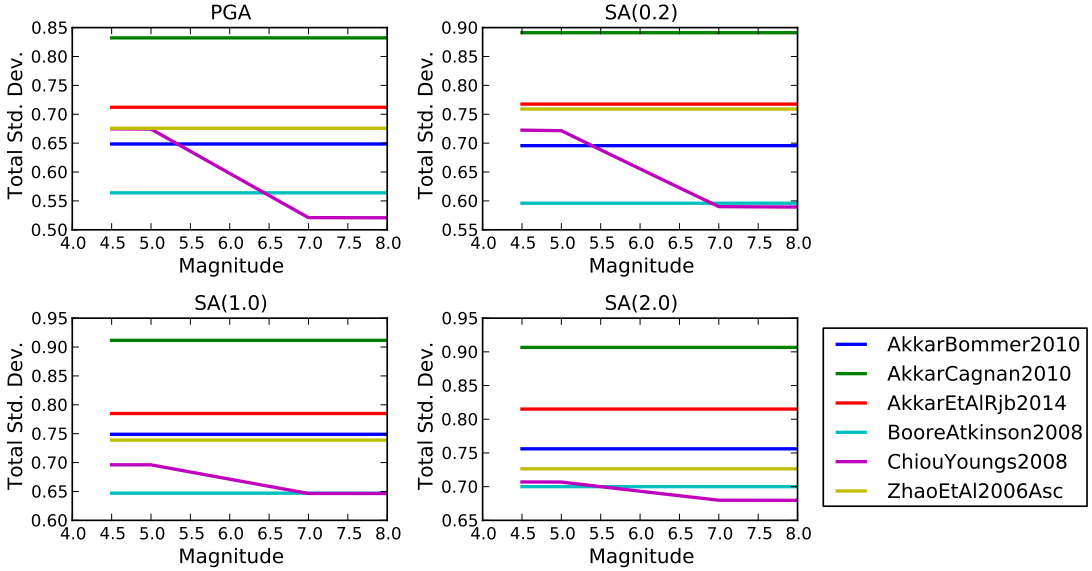


Figure 4.2 – *Scaling of total standard deviation of the GMPEs with respect to magnitude*

Figure 4.2 shows the scaling of the total standard deviation with magnitude. The type of standard deviation is configured using the `stddevs` option, which can take the values: "`Total`", "`Inter event`" or "`Intra event`". Examples of inter- and intra-event standard deviation are shown in Figure 4.3 and 4.4 respectively.

4.2.2 Comparing the Scaling with Distance

With this tool it is possible to compare how the GMPEs describe the attenuation of strong motion with distance for a given magnitude. As before, in order to utilise the tool in a "non-specific rupture" context, it is necessary to determine the distances relations manually. In this case we consider the same vertical rupture with a top of rupture depth of 5 km. For convenience we assume that the line of attenuation is perpendicular to the rupture strike, originating at the epicentre:

```
magnitude = 6.5
distances = {"repi": np.arange(0.0, 151.0, 1.0)}
distances["rhypo"] = np.sqrt(distances["repi"] ** 2.0 +
                             params["hypo_depth"] ** 2.0)
distances["rjb"] = distances["repi"]
distances["rrup"] = np.sqrt(distances["rjb"] ** 2.0 +
                            params["ztor"] ** 2.0)
distances["rx"] = distances["rjb"]
```

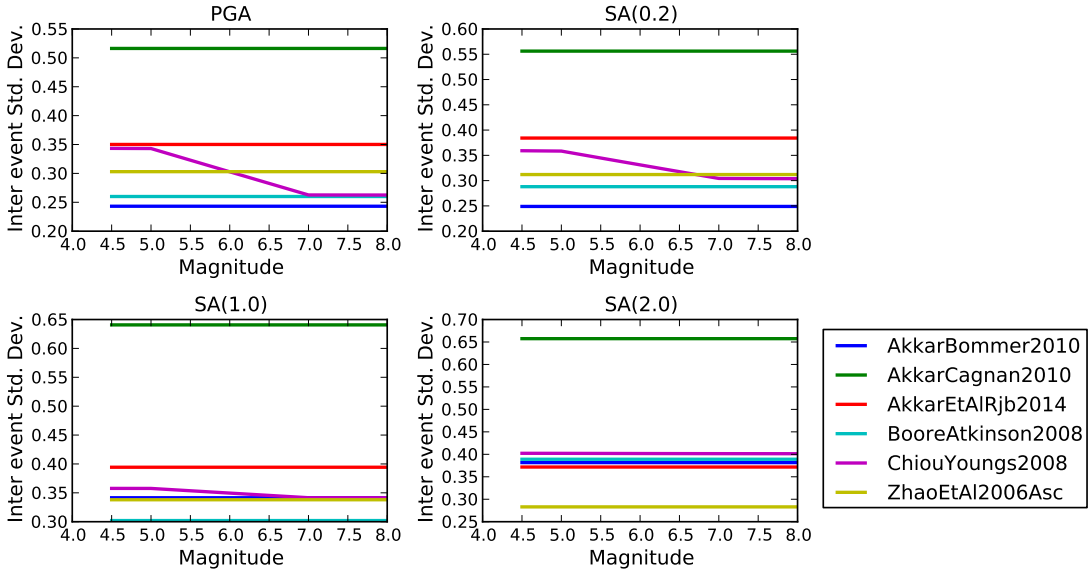


Figure 4.3 – *Scaling of inter-event deviation of the GMPEs with respect to magnitude*

The distance trellis plots, as shown in Figure 4.5 can be created with the following command:

```
trpl.DistanceIMTTrellis(magnitude,
                        distances,
                        gmpe_list,
                        imts,
                        params,
                        distance_type="rjb",
                        plot_type="loglog",
                        filename="path/to/image",
                        filetype="png")
```

The inputs are similar to that of the `trpl.MagnitudeIMTTrellis` tool, except that the user has the option of specifying which distance measure to use on the abscissa of the plot via the `distance_type` keyword. The use can also choose whether to plot the distance in logarithmic or linear axes via the `plot_type` keyword, which would take the value of "loglog" or "semilogy" respectively. The same GMPEs plotted in terms of linear distance axes are shown in Figure 4.6

As with the magnitude trellis plots. The standard deviations can also be plotted, as is the case for the total standard deviation shown in Figure 4.7:

```
trpl.DistanceSigmaIMTTrellis(magnitude,
                             distances,
                             gmpe_list,
                             imts,
                             params,
                             stddevs="Total"
                             distance_type="rjb",
                             plot_type="loglog",
                             filename="path/to/image",
                             filetype="png")
```

4.2.3 Comparing the Scaling of the Response Spectrum with Magnitude and Distance

Whilst understanding the scaling of a particular IMT can be useful, it can also be insightful to understand how a GMPE will scale the response spectrum with magnitude and distance. In the

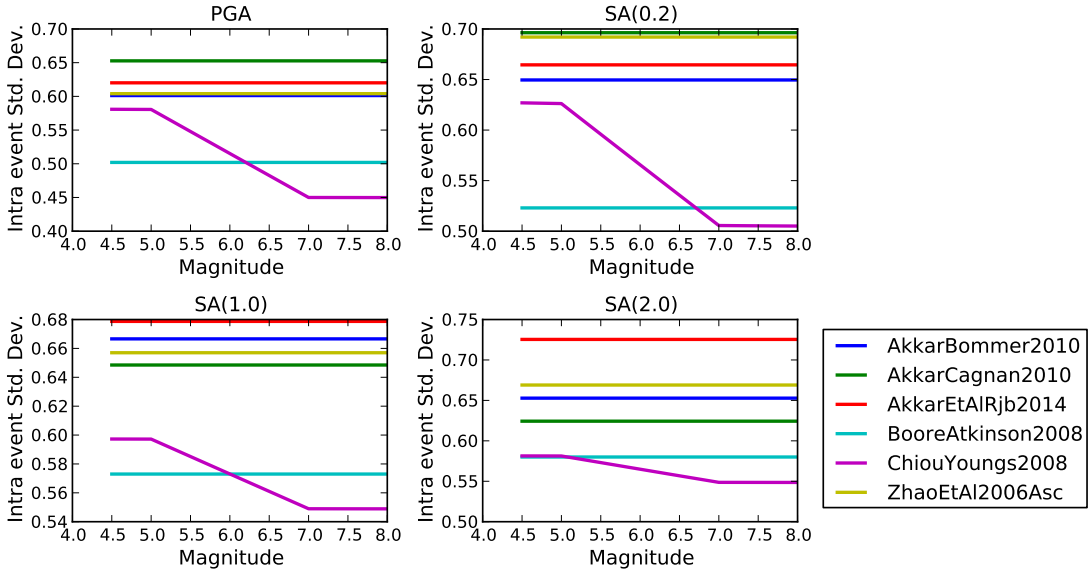


Figure 4.4 – *Scaling of intra-event deviation of the GMPEs with respect to magnitude*

following example we consider four magnitudes: M_W 5.0, M_W 6.0, M_W 7.0 and M_W 8.0; and four source-site distances R_{JB} 5.0, R_{JB} 20.0, R_{JB} 50.0, R_{JB} 100.0. Once again it is necessary to calculate the geometry in this simple case:

```
# Choose 4 magnitudes for comparison: 5, 6, 7, 8
magnitudes = np.array([5.0, 6.0, 7.0, 8.0])
# Choose 4 distances for comparison: 5., 20., 50., 100.
distances = {"repi": np.array([5.0, 20.0, 50.0, 100.0])}
distances["rhypo"] = np.sqrt(distances["repi"] ** 2.0 +
                             params["hypo_depth"] ** 2)
distances["rjb"] = distances["repi"]
distances["rrup"] = np.sqrt(distances["rjb"] ** 2.0 +
                            params["ztor"] ** 2)
distances["rx"] = distances["rjb"]
```

We need also to specify the spectral periods we wish to use for calculation. In this case, as citeAkkarCagnan2010 is limited to just 2 s period, we limit the period range for comparison so 0.05 s to 2 s.

```
periods = [0.05, 0.075, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15,
           0.16, 0.17, 0.18, 0.19, 0.20, 0.22, 0.24, 0.26,
           0.28, 0.30, 0.32, 0.34, 0.36, 0.38, 0.40, 0.42,
           0.44, 0.46, 0.48, 0.50, 0.55, 0.60, 0.65, 0.70,
           0.75, 0.80, 0.85, 0.90, 0.95, 1.00, 1.10, 1.20,
           1.30, 1.40, 1.50, 1.60, 1.70, 1.80, 1.90, 2.00]
```

To produce the comparison shown in Figure 4.8 simply run:

```
trpl.MagnitudeDistanceSpectraTrellis(magnitudes,
                                     distances,
                                     gmpe_list,
                                     periods,
                                     params,
                                     plot_type="loglog",
                                     filename="path/to/image",
                                     filetype="png")
```

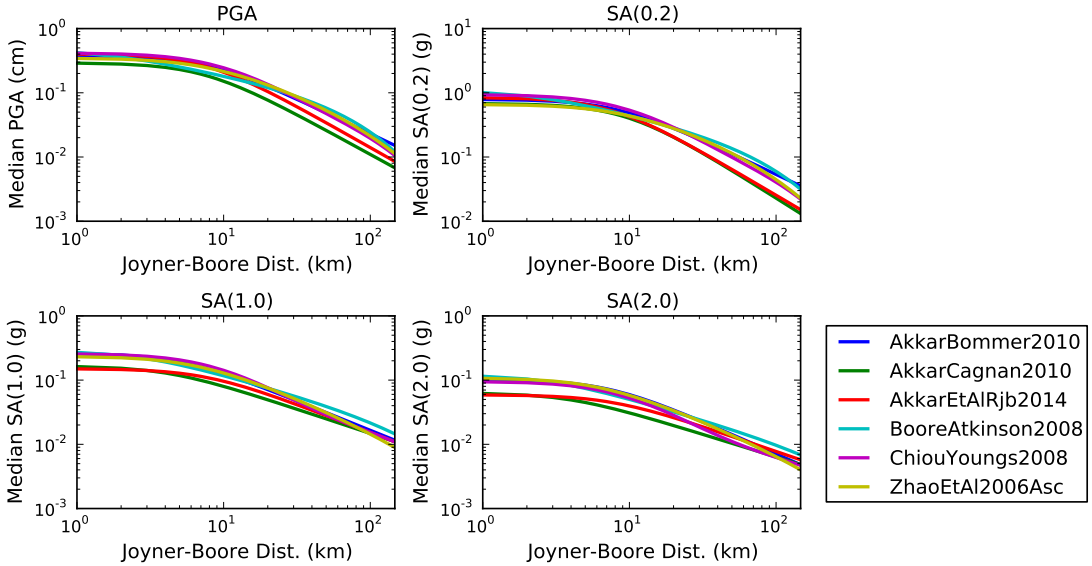


Figure 4.5 – *Scaling of the GMPEs with respect to distance*

Once again the corresponding standard deviations, shown in Figure 4.9, can also be plotted:

```
trpl.MagnitudeDistanceSpectraSigmaTrellis(magnitudes,
                                          distances,
                                          gmpe_list,
                                          periods,
                                          params,
                                          plot_type="semilogy",
                                          figure_size=(10,8),
                                          filename="path/to/image",
                                          filetype="png")
```

4.3 Trellis Plotting: The Simple Way

The examples seen section 4.2 are relatively feasible to implement in part because we are assuming a conveniently simple rupture configuration. To really understand how the GMPEs may compare in more physically consistent applications it is preferable to render the comparison with respect to a more realistic rupture alignment. To do this the GMPE-SMTK supplied a set of tools to help set up a rupture model that can be used to form the basis for the comparisons. These additional tools can be found in the `trellis/configure.py` module, which can be imported thus:

```
import smtk.trellis.configure as rcfg
```

A rupture is represented as an instance of the `rcfg.GSIMRupture` class, which can be created with the following essential information:

- `magnitude` The moment magnitude of the rupture
- `dip` The dip (in degrees) of the rupture
- `aspect` The along-strike to down-dip length ratio of the rupture

Further parameters can also be configured:

- `rake` Rake (degrees) of the rupture, default of 0.0
- `ztor` Top of rupture depth (km), default to 0 km
- `strike` Strike of the rupture (km, default to 0.0

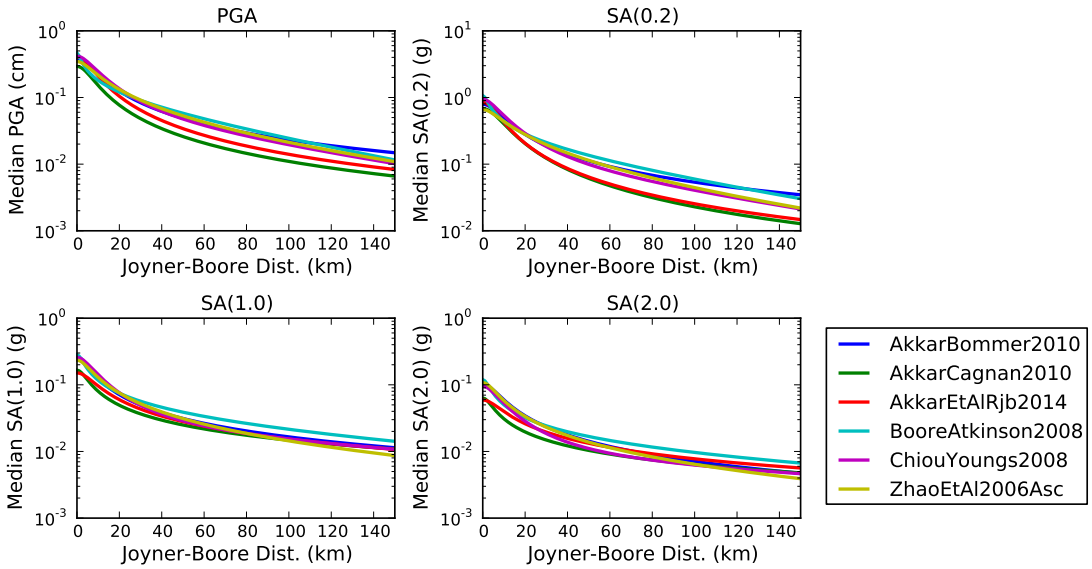


Figure 4.6 – *As Figure 4.5 with linear distance axes*

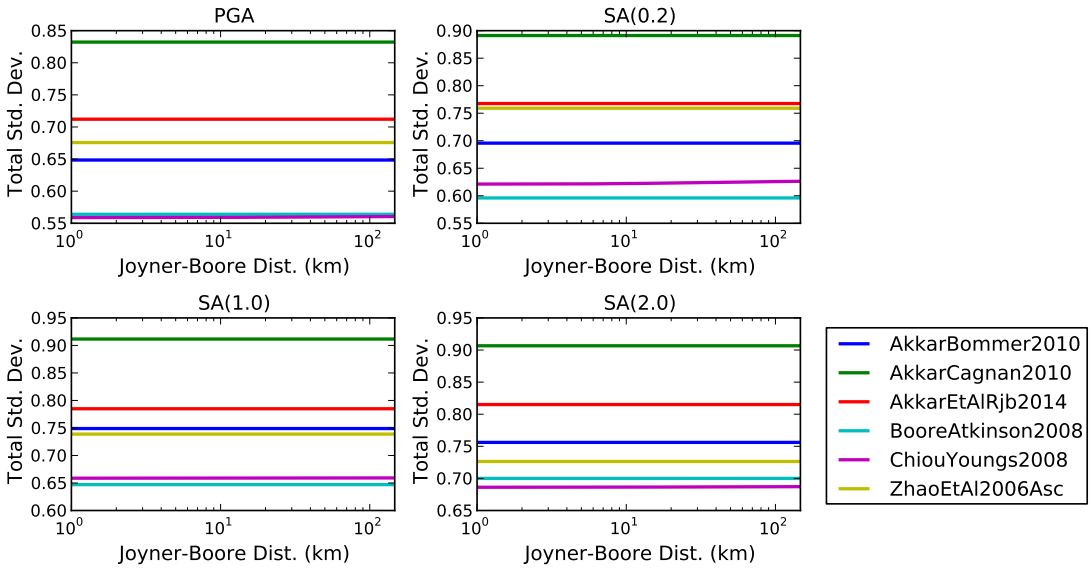


Figure 4.7 – *Scaling of the total standard deviation with distance.*

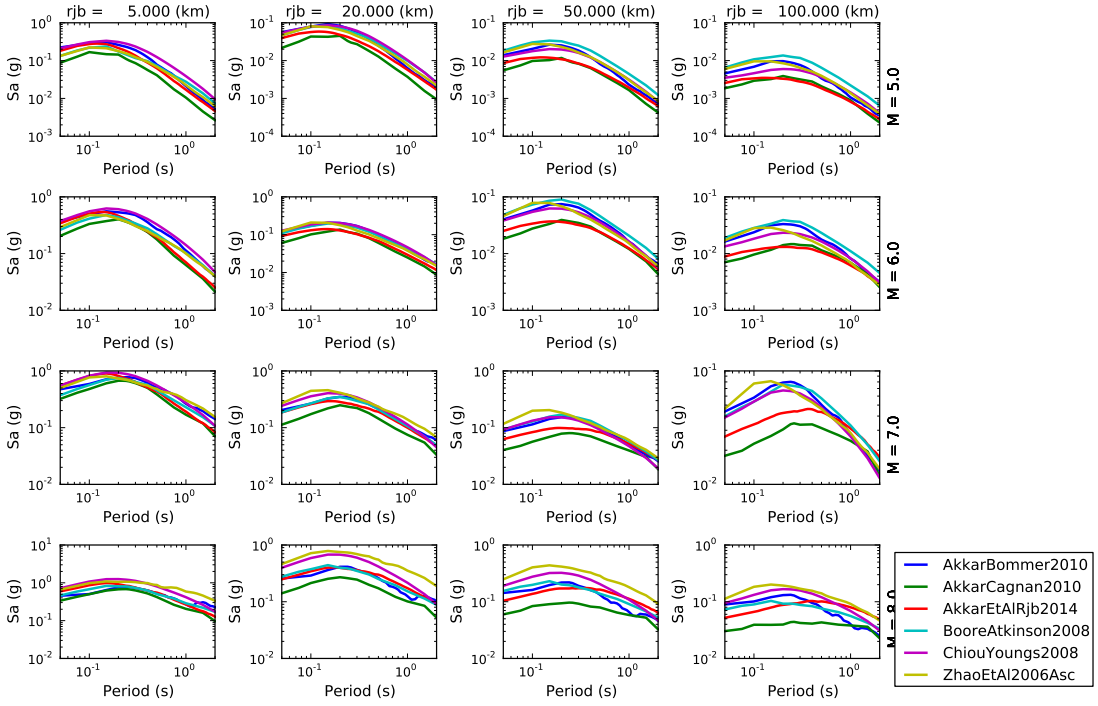


Figure 4.8 – *Scaling of the response spectrum for each GMPE for selected magnitude and source-site distance combinations*

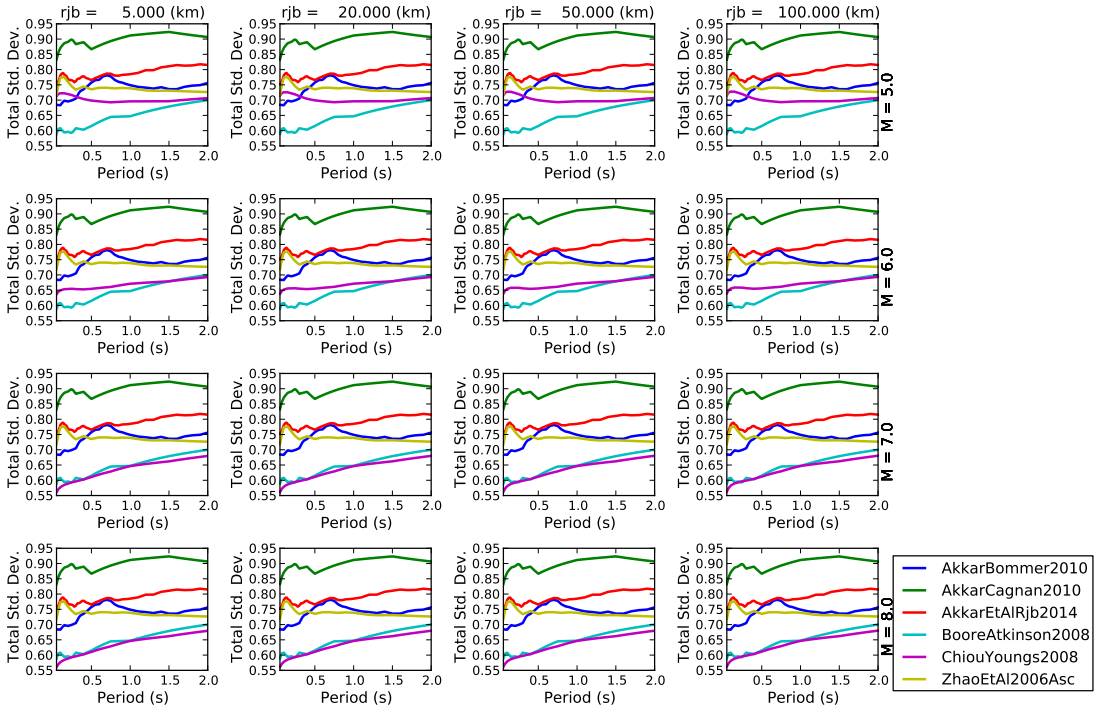


Figure 4.9 – *Scaling of the response spectrum total standard deviation for each GMPE for selected magnitude and source-site distance combinations*

- `msr` Magnitude scaling relation, default to Wells and Coppersmith, 1994
- `initial_point` Location of the centroid on the Earth's surface as instance of `openquake.hazardlib.geo.point.Point` class
- `hypocentre_location` The location of the hypocentre within the rupture plane as a tuple of fraction of along-strike length and fraction of down-dip width (e.g. to place the hypocentre in the centroid of the rupture set this to `(0.5, 0.5)`, the default value)

In the following example we consider a rupture plane from an earthquake of magnitude M_W 6.5, dip of 45°, aspect ratio of 1.5, rake of 90° (i.e. reverse fault), strike of 0° and hypocentre located in the centroid of the rupture:

```
rupt1 = rcfg.GSIMRupture(magnitude=6.5,
                         dip=45.0,
                         aspect=1.5,
                         rake=90.0,
                         ztor=0.0,
                         strike=0.0,
                         hypocentre_location=(0.5, 0.5))
```

4.3.1 Configuring the Sites

In order to generate the trellis plots we need to generate a site configuration that permits for the different distances to be rendered. There are three options by which this can be achieved, the selection of which may depend on the application.

Site as a Mesh

The first option, and the most computationally intensive, is to generate the target sites as an evenly spaced mesh of points centred on the rupture. This can be done using the method of the `rcfg.GSIMRupture` class named `get_target_sites_mesh`:

```
sites_mesh = rupt1.get_target_sites_mesh(maximum_distance=200.0,
                                         spacing=2.0,
                                         vs30=800.0,
                                         vs30measured=True,
                                         z1pt0=None,
                                         z2pt5=None)
```

Here the keyword `maximum_distance` specifies the maximum Joyner-Boore distance from the rupture to extend the mesh, `spacing` defines the mesh spacing (km), `vs30` defines the 30-m averaged shear-wave velocity to assign to all the sites, `vs30measured` is a boolean parameter indicating whether the `vs30` is measured (True, default) or inferred (False), `z1pt0` and `z2pt5` are the depth to the 1 km/s and 2.5 km/s shearwave velocity interfaces respectively. These are optional but if not specified they will be calculated from the `vs30` value using the models of **AbrahamsonSilva2008** and **CampbellBozorgnia2008** respectively.

To visualise the rupture and site configuration as any time, simply use the `plot_model_configuration` method found in the rupture class:

```
rupt1.plot_model_configuration(figure_size=(7,5),
                               filename="/path/to/image",
                               filetype="png",
                               dpi=300)
```

The rupture-site mesh configuration for the above settings is shown in Figure 4.10

The `rcfg.GSIMRupture` class contains an additional method to allow the user to compare the distance metrics for the particular rupture and site configuration. For example, to compare the Joyner-Boore distance and rupture distance for the rupture mesh shown in Figure 4.10, the following will produce the plot shown in Figure 4.11

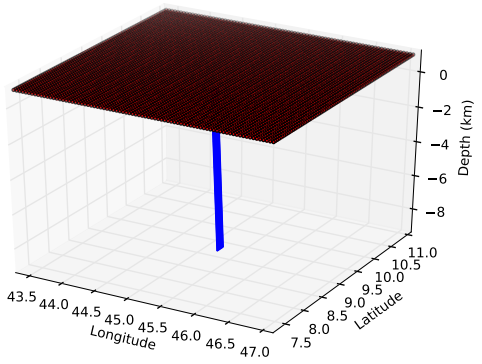


Figure 4.10 – *Rupture and site configuration for a regularly spaced mesh of points around the rupture*

```
rupt1.plot_distance_comparisons("rjb",
                                "rrup",
                                filename="path/to/image",
                                filetype="pdf")
```

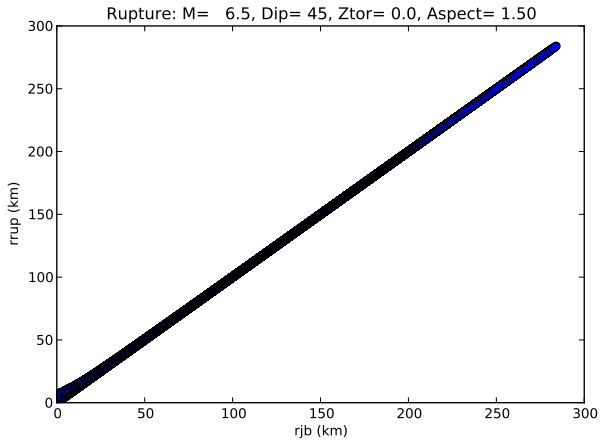


Figure 4.11 – *Comparison of Rupture distance with Joyner-Boore distance for the rupture and site configuration in Figure 4.10*

Sites as a Line

A more common configuration for exploring the attenuation properties of the GMPEs is to configure the sites in a single line propagating away from the rupture. However, it is important to recognise that the relation between the distance metrics will depend on the azimuth of the line with respect to the strike of the rupture. These properties can be configured using the command below, and the corresponding plot shown in Figure 4.12:

```
sites_line = rupt1.get_target_sites_mesh(maximum_distance=200.0,
                                         spacing=1.0,
                                         vs30=800.0,
                                         line_azimuth=90.
```

```
                                                vs30measured=True,
                                                z1pt0=None,
                                                z2pt5=None)
rupt1.plot_model_configuration()
```

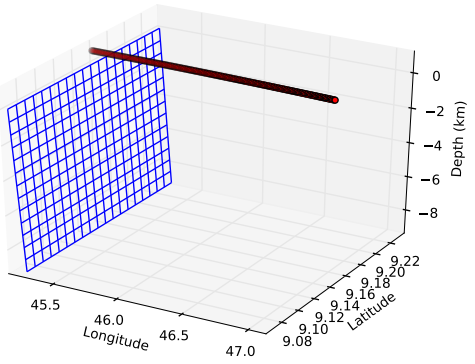


Figure 4.12 – *Rupture and site configuration for a regularly spaced line or points propagating at an angle of 90° from the rupture, along the hanging wall*

Site as a Point

When comparing GMPEs in terms of spectra it is preferable to consider the target site as a single point. In this case this both the distance, the distance type and the azimuth must be specified, in addition to the information required for the other site configurations. So to consider a site located at a Joyner-Boore distance of 20 km on a line perpendicular to the rupture, on the hanging wall, the configuration shown in Figure 4.13 can be constructed via:

```
sites_line = rupt1.get_target_sites_point(distance=20.0,
                                          distance_type="rjb",
                                          vs30=800.0,
                                          line_azimuth=90.
                                          vs30measured=True,
                                          z1pt0=None,
                                          z2pt5=None)
rupt1.plot_model_configuration()
```

4.3.2 Creating the Trellis Points

Each of the trellis plotting tools shown in section 4.2 can be implemented using their own built-in method called `from_rupture_model`. When using this method rather, than requiring the user manually specify the magnitude, distance and parameter inputs, it is possible to instead input only the rupture model.

For a rupture-specific configuration, distance trellis plots similar to those shown in Figure 4.12 can be created as follows:

```
trpl.DistanceIMTTrellis.from_rupture_model(
    rupt1,
    gmpe_list,
    imts,
    filename="path/to/image",
    filetype="pdf")
```

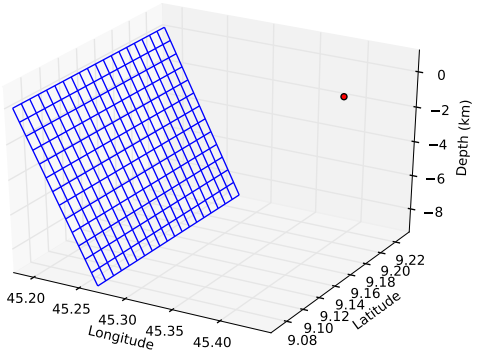


Figure 4.13 – *Rupture and site configuration for a single point located at 20 km (Joyner-Boore) from the rupture on a bearing of 90° on the hanging wall*

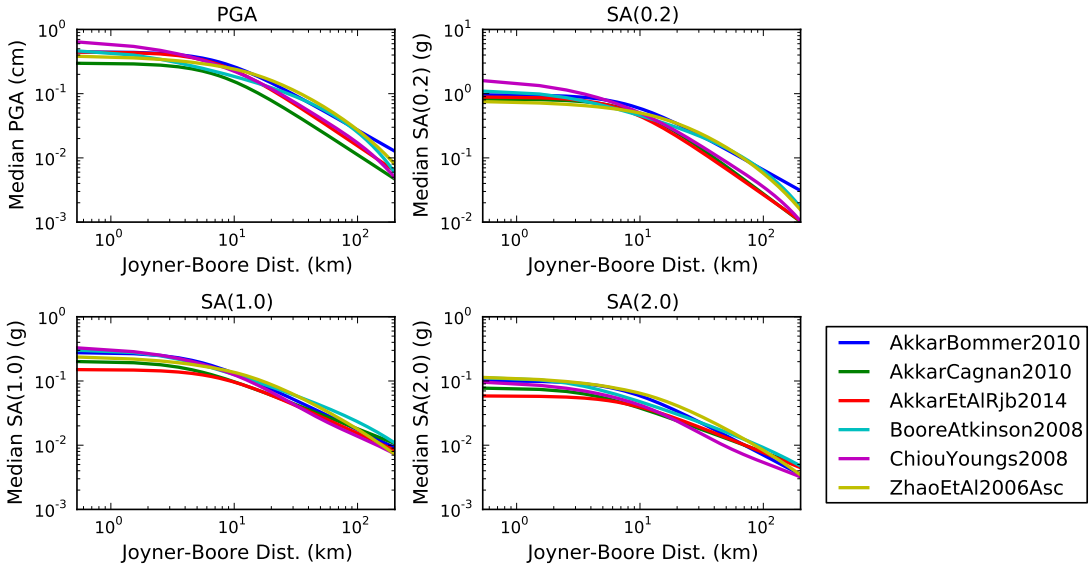


Figure 4.14 – *Scaling of the GMPEs with respect to distance for the rupture and site configuration in Figure 4.12*

Likewise, the standard deviation plots (e.g. Figure 4.15) can be created in a similar manner:

```
trpl.DistanceSigmaIMTTrellis.from_rupture_model(
    rupt1,
    gmpe_list,
    imts,
    filename="path/to/image",
    filetype="pdf")
```

The rupture-specific configuration allows for the same comparison to take place on the footwall of the rupture, for those GMPE's such as citeChiouYoungs2008 that have hanging wall coefficients. For example, rather than propagating the line at 90° from the rupture on the hanging wall, we will place the line at an azimuth of 230° from the rupture (i.e. on the footwall):

```
sites_line = rupt1.get_target_sites_mesh(maximum_distance=200.0,
                                         spacing=1.0,
```

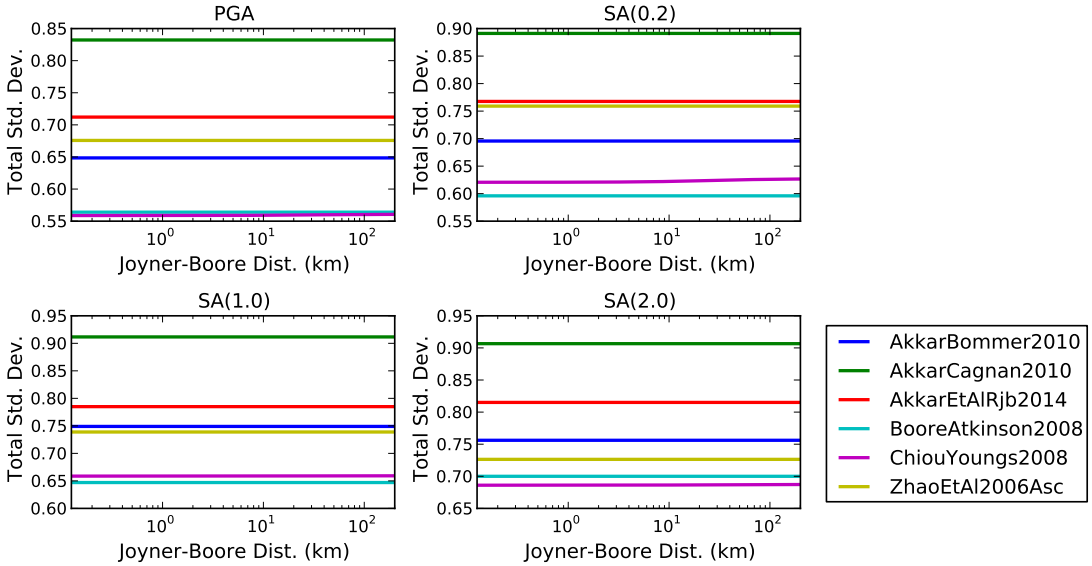


Figure 4.15 – *Scaling of the total standard deviation GMPEs with respect to distance for the rupture and site configuration in Figure 4.12*

```
3                                                   vs30=800.0,
4                                                   line_azimuth=230.
5                                                   vs30measured=True,
6                                                   z1pt0=None,
7                                                   z2pt5=None)
8  rupt1.plot_model_configuration()
```

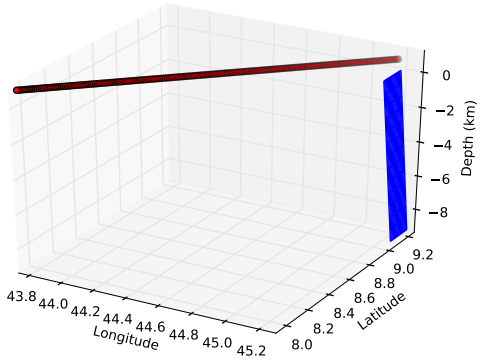


Figure 4.16 – *Rupture and site configuration for a regularly spaced line or points propagating at an angle of 90° from the rupture, along the footwall*

The corresponding trellis plots for the expected ground motion values is shown in Figure 4.17

This same comparison can be undertaken for the response spectra too. In this case we consider the rupture and site configuration shown in figure 4.13 (site at 20 km R_{JB} from the rupture along a bearing of 90°). To view the predicted response spectra at this site it is only necessary to do the following to produce the plots shown in Figures 4.18 and 4.19:

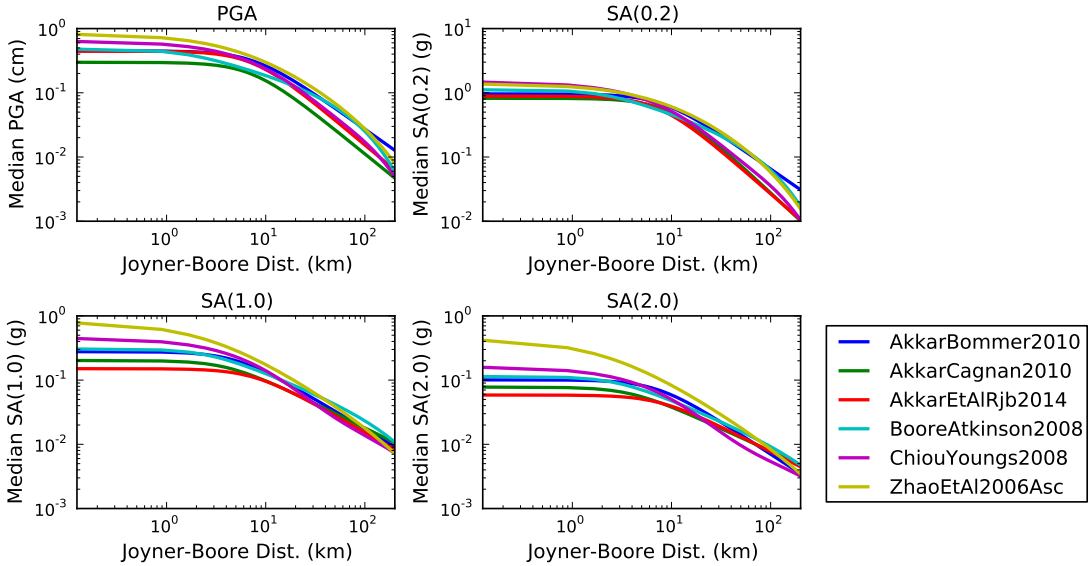


Figure 4.17 – *Scaling of the GMPEs along the footwall of a rupture, configured according to Figure 4.16*

```
sites_line = rupt1.get_target_sites_point(distance=20.0,
                                          distance_type="rjb",
                                          vs30=800.0,
                                          line_azimuth=90.
                                          vs30measured=True,
                                          z1pt0=None,
                                          z2pt5=None)
# Create the expected ground motion trellis plot
trpl.MagnitudeDistanceSpectraTrellis.from_rupture_model(
    rupt1,
    gmpe_list,
    periods,
    filename="path/to/image",
    filetype="png")
trpl.MagnitudeDistanceSpectraSigmaTrellis.from_rupture_model(
    rupt1,
    gmpe_list,
    periods,
    stddevs="Total",
    filename="path/to/image",
    filetype="png")
```

The same comparison can also be done on the footwall of the rupture. If the site is now placed at a Joyner-Boore distance of 20 km on the at a bearing of 220° from the rupture (i.e. on the footwall), the configuration in Figure 4.20 is created as shown:

```
sites_line = rupt1.get_target_sites_point(distance=20.0,
                                          distance_type="rjb",
                                          vs30=800.0,
                                          line_azimuth=220.
                                          vs30measured=True,
                                          z1pt0=None,
                                          z2pt5=None)
rupt1.plot_model_configuration()
```

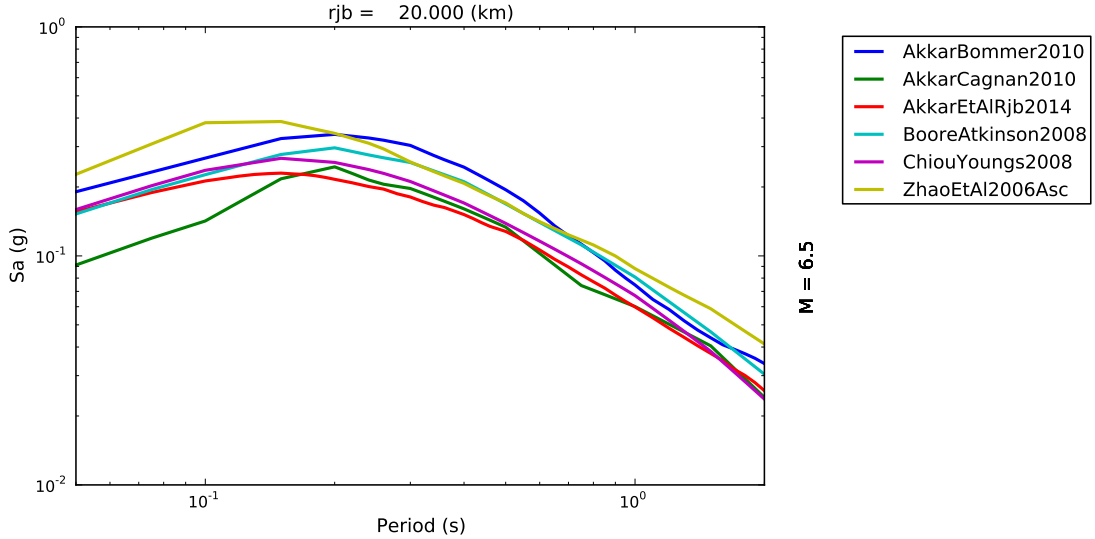


Figure 4.18 – *Comparison of response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.13*

The corresponding GMPEs and their respective total standard deviations are shown in Figures 4.21 and 4.22 respectively.

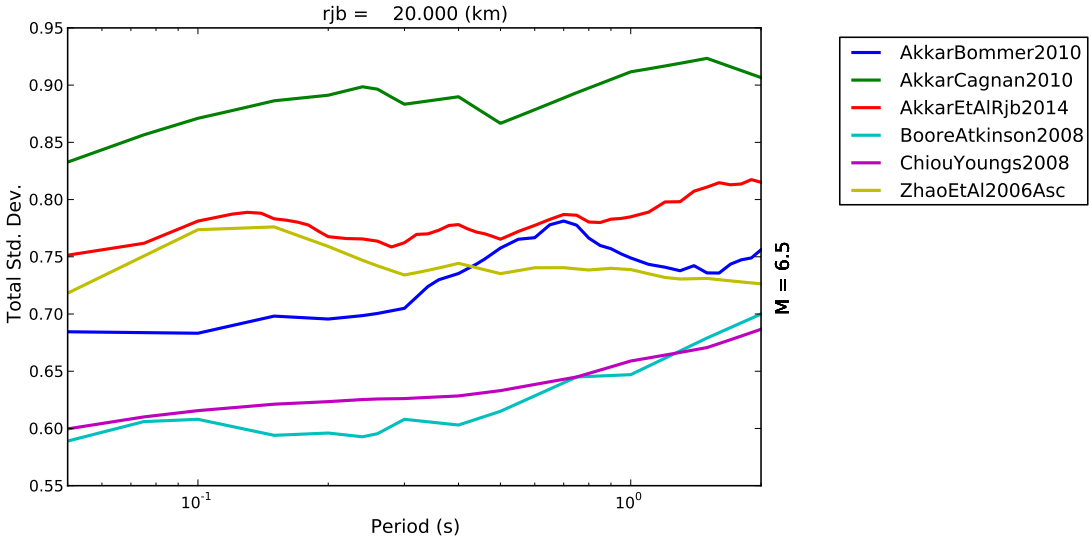


Figure 4.19 – *Comparison of total standard deviation of the response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.13*

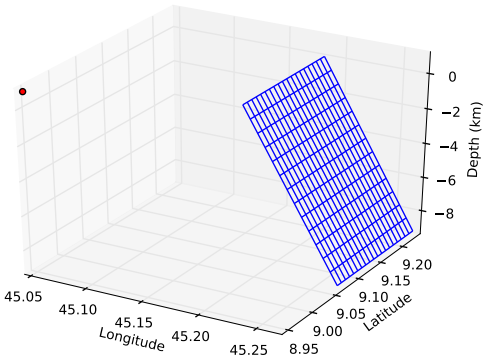


Figure 4.20 – *Rupture and site configuration for a single point located at 20 km (Joyner-Boore) from the rupture on a bearing of 220° on the hanging wall*

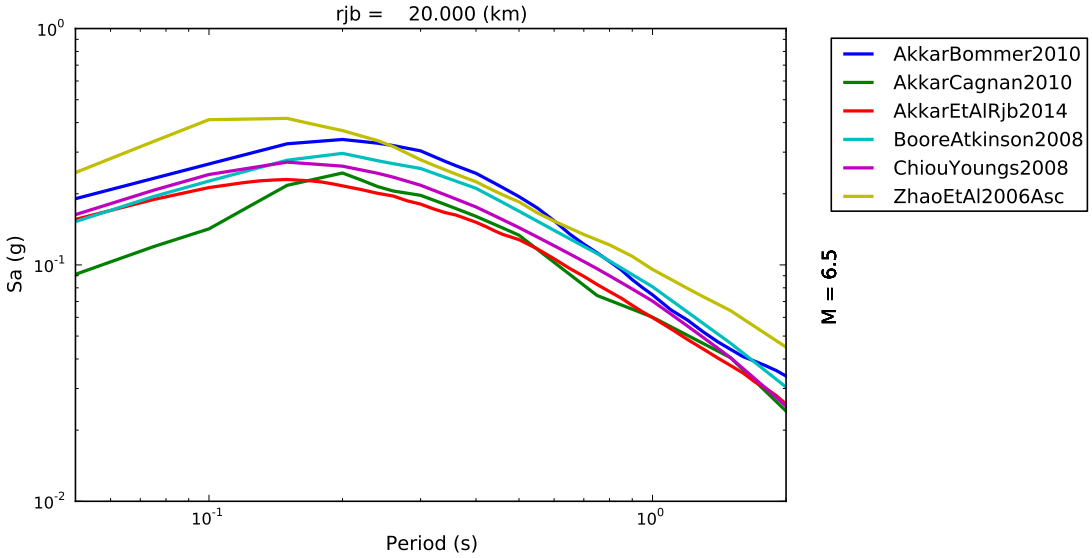


Figure 4.21 – *Comparison of response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.20*

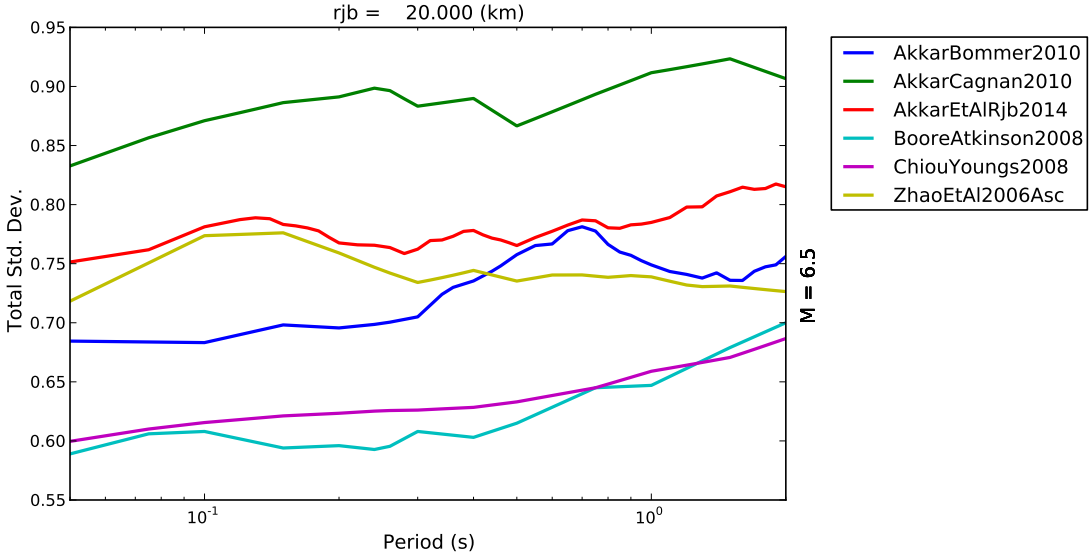


Figure 4.22 – *Comparison of total standard deviation of the response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.20*

5. Comparing Models and Observations

5.1 GMPE Residuals

5.1.1 Definition of the GMPE Residuals

Arguably one of the most critical processes in GMPE selection, and GMPE weighting in an epistemic uncertainty analysis, is the comparison of the GMPEs against set of ground motions for the region of application. These analyses can serve several purposes. The first is to understand the extent to which the GMPE of interest can represents the local properties of source, path and site scaling of the ground motion. In modelling both the expected ground motion and the aleatory varibility, each GMPE is a probability distribution in which a ground motion y_{ij} at recorded location j, originating from event i, is characteristed by a lognormal distribution:

$$\log y_{ij} = \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right) + z_{T,ij}\sigma_T \tag{5.1}$$

where $\mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right)$ is the expected ground motion from an event of magnitude m_i, recorded at a distance r_{ij}, where $\theta_{\mathbf{ij}}$ corresponds to various other model parameters relevant to the model in question (e.g., site amplification, basin response, hanging wall scaling etc.). The total uncertainty $(z_{T,ij})$ is therefore modelled as a normal distribution with a mean of zero and a standard deviation of σ_T. The term $z_{T,ij}$ is therefore the "total" normalised residual of the j^{th} recording from the i^{th} event:

$$z_{T,ij} = \frac{\log\left(y_{ij}\right) - \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right)}{\sigma_T} \tag{5.2}$$

For most GMPEs, however, the a random effects regression process is used to derive the coefficients of the model. This divides the total residual term into an inter-event component $\delta_{E,i}$ and an the intra-event component $\delta_{A,ij}$, such that equation can be written as:

$$\log y_{ij} = \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right) + \delta_{E,i} + \delta_{A,ij} \tag{5.3}$$

The inter-event residual is normally distributed with a mean of zero and a standard deviation of τ, and likewise the intra-event residual is normally distributed with a zero mean and a standard

deviation of σ. Consequently equation 5.3 can be finally written as:

$$\log y_{ij} = \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right) + z_{E,i}\tau + z_{A,ij}\sigma \tag{5.4}$$

Where $z_{E,i}$ and z_{A_ij} are the normalised inter- and intra-event residuals respectively. Following the random effects definition of citeAbrhamsonYoungs1992, the inter-event term is defined via:

$$\delta_{E,i} = z_{E,i}\tau = \frac{\tau^2 \sum_{j=1}^{n_i}\left(y_{ij} - \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right)\right)}{n_i\tau^2 + \sigma^2} \tag{5.5}$$

where n_i is the number of records observed from event i. The intra-event residual the follows:

$$\delta_{A,ij} = z_{A,ij}\sigma = \log y_{ij} - \mu\left(m_i, r_{ij}, \theta_{\mathbf{ij}}\right) - z_{E,i}\tau \tag{5.6}$$

5.1.2 Residual analysis in the GMPE-SMTK

The GMPE-SMTK offers the ability to compare observed ground motions with the GMPE model predictions using the GMPE implementations found inside OpenQuake. This is a particularly powerful tool as it permits, possibly for the first time, a hazard modeller to undertake the GMPE comparison using the exact same GMPE implementation found inside the seismic hazard software. This ensures consistency between, for example, the definitions of the distance metrics, as well as allowing the GMPE-SMTK to inherit the comprehensive test coverage that is fundamental to the OpenQuake software.

Several key sets of tools can be found within the GMPE-SMTK to allow the modeller not only to explore particular features of the GMPE residuals, but also to undertake quantitative analysis of the overall fit of each GMPE to observed strong motion data. All the analyses undertaken will, by default, analyse the total residual, the inter-event residual and the intra-event residual, following the approach described in cite(StaffordEtAl2008). If the GMPE in question does not provide coefficients to define the inter- and intra-event residual only the total residual term will be considered.

The GMPE-SMTK provides two sets of tools. The first tool set contains the tools to undertake the analysis of residuals, and/or the model fit. These tools are found in the module `smtk.residuals.gmpe_residuals`. The second set will provide the visualisation and plotting functionalities. These are found in the `smtk.residuals.residual_plotter` module. In the following examples we will refer to these modules as `res` and `rspl` respectively, and they can be imported via:

```
1 import smtk.residuals.gmpe_residuals as res
2 import smtk.residuals.residual_plotter as rspl
```

To generate a set of ground motion residuals the tools require three pieces of information: a ground motion database (as an instance of the `smtk.sm_database.GroundMotionDatabase` class), a list of the required GMPEs and a list of the required intensity measures. In the following examples we consider a set of strong motion records recorded in Italy and compare four GMPEs (BooreAtkinson2008, AkkarBommer2010, AkkarCagnan2010, AkkarEtAl2014) and four intensity measures (PGA, $Sa\,(0.2s)$, $Sa\,(1.0s)$, $Sa\,(2.0s)$). Typically, these will be set-up as follows:

```
1 # Import CPickle
2 import cPickle
```

```
# Import the database
sm_database = cPickle.loads(open("path/to/metadata.pkl", "r"))
# Define the list of GMPEs
gmpe_list = ["BooreAtkinson2008",
             "AkkarBommer2010",
             "AkkarCagnan2010",
             "AkkarEtAlRjb2014"]
# Define the list of intensity measures
imt_list = ["PGA", "SA(0.2)", "SA(1.0)", "SA(2.0)"]
```

For undertaking simple analysis of residuals simply run:

```
# Set-up the residual calculator
resid1 = res.Residuals(gmpe_list, imts)
# Run the residual calculator
resid1.get_residuals(sm_database)
```

When run, the class `Residuals` will hold an attribute `residuals`, which will contain a set of nested python dictionaries in which the corresponding residual values are stored. To retrieve the residual values for a specific GMPE and intensity measure then type:

```
res_data = resid1.residuals["GMPE Name"]["IM Name"]["Type"]
```

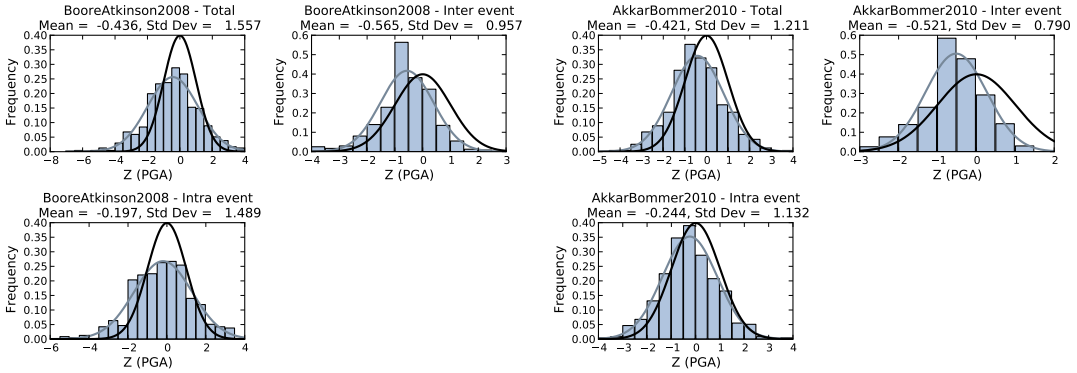
where, "GMPE Name" is the name of the GMPE specified in the input list, "IM Name" is the name of the intensity measure (as specified in the input list) and "Type" is one of "Total", "Inter event" or "Intra event" (note the syntax - the keys are case sensitive). The residuals will then be a single vector of residual values for that specific GMPE, IM and residual type.

5.2 Basic Trends in Residuals

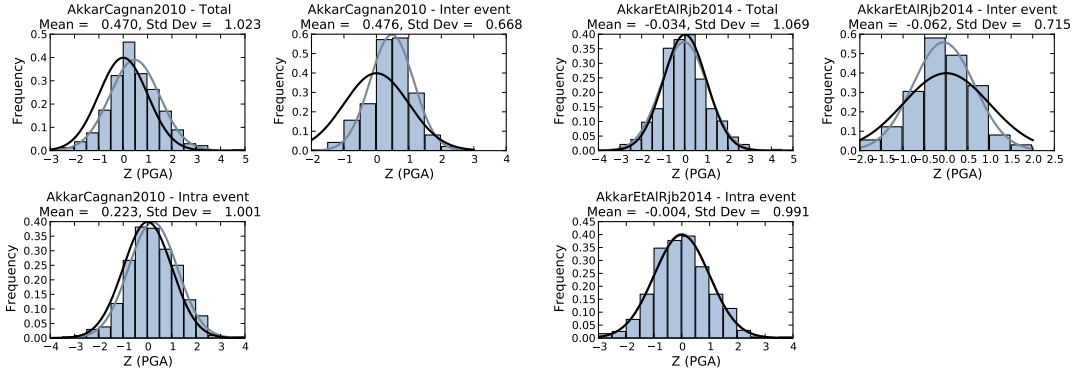
One of the simplest means of comparing observations with the models is via the use of histograms of the residual terms z_T, $z_{E,i}$ and $z_A ij$ from the observed records with respect to the GMPE. These help identify the general tendency of the GMPE with respect to the records. From the definitions described in 5.1 it follows that for a GMPE to represent a good fit to a set of records each of the normalised residual distributions should match closely to a standard normal distribution (i.e with a mean of zero and standard deviation of 1.0). Differences in the mean of the residuals may indicate a tendency for the GMPE to predict higher or lower values than those observed in records, whilst differences in the standard deviation may suggest an over- or underestimation of the variability of the ground motions.

Basic histograms of the residuals of a GMPE can be created using the `rspl` tool `ResidualPlot`. For the GMPEs listed in previously the following command will produce the four sets of histogram plots for the trend in PGA residuals shown in Figure 5.1 and, for $Sa\,(1.0s)$ using Akkar et al (2014) only in Figure 5.2.

```
# Boore & Atkinson (2008)
rspl.ResidualPlot(resid1, "BooreAtkinson2008", "PGA",
                  filename="path/to/image.png",
                  filetype="png")
# Akkar & Bommer (2010)
rspl.ResidualPlot(resid1, "AkkarBommer2010", "PGA")
# Akkar & Cagnan (2010)
rspl.ResidualPlot(resid1, "AkkarCagnan2010", "PGA")
# Akker et al. (2014)
rspl.ResidualPlot(resid1, "AkkarEtAlRjb2014", "PGA")
# Akker et al. (2014) - Sa (1.0 s)
rspl.ResidualPlot(resid1, "AkkarEtAlRjb2014", "Sa(1.0)")
```



(a) *citeBooreAtkinson2008 model* **(b)** *citeAkkarBommer2010 model*



(c) *citeAkkarCagnan2010 model* **(d)** *citeAkkarEtAl2014 model*

Figure 5.1 – *PGA residual distribution for the record set. The grey line indicates the probability density function fit to the data, whilst the black line indicates the standard normal density function*

Figures 5.1 and 5.2 shows a relatively discernible set of trends. Consider just the Boore and Atkinson, 2008 model for PGA (Figure 5.1a). Each of the residuals have a mean less than zero, significantly so in the case of the inter-event residual. This indicates that in general the Boore and Atkinson, 2008 model is predicts higher ground motions than those observed in the records and most of this can be attributed to the inter-event term. The distributions also show a greater spread than expected from the model, as demonstrated by the standard deviation greater than unity. In this case it is the intra-event residuals that show a greater spread than expected, whilst spread in the inter-event residuals is in better agreement with the model. In contrast the Akkar et al. (2014) GMPE (from which the current records form part of the generating data set) show a much closer agreement with the records (as expected). It is seen that that the distribution of the normalised total residuals is very close to the standard normal distribution; a trend that is matched in the intra-event residuals. Similarly in the inter-event residuals the mean value is close to zero, and only the spread of the model shows a difference, in this case showing a smaller variance than expected from the model. This is to be expected as the current dataset is based on exclusively Italian records (predominantly extensional events, with only a few thrust or oblique slip records), whereas the GMPE was fit to a full set of European and Middle Eastern records, thus incorporating a greater variability of mechanism types than just the Italian subset. The same trend can be seen in the residuals for $Sa\,(1.0s)$.

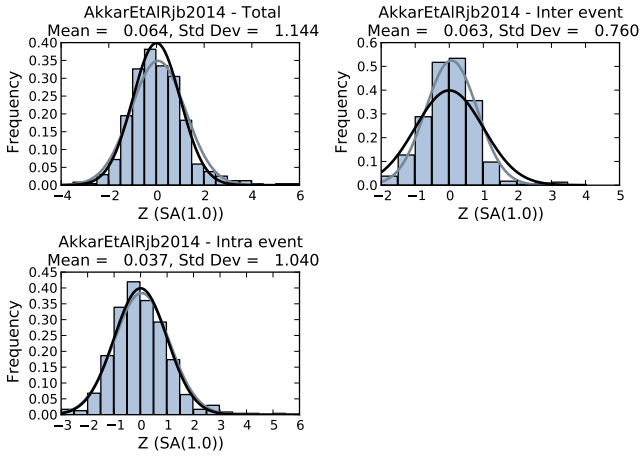


Figure 5.2 – *PGA residual distribution for the record set using the Akkar et al. (2014) GMPE*

5.3 The Likelihood Model (TOCITE Scherbaum et al., 2004)

The basic residual trends can help understand where biases may be present in the model with respect to the record set, but they do not necessarily provide a quantitative indication of the total fit of the model to the record. One possibility to do this (others will presented later in the chapter) is to consider the overall goodness-of-fit of the model to the predictions. SCHERBAUM ETAL 2004 consider the probability that the absolute value of a random sample from the normalised distribution to fall into the interval between the modulus of a particular observation $|z_0|$ and infinity.

$$u(z_0) = \frac{1}{2} Erf\left(\frac{z_0}{\sqrt{2}}, \infty\right) \tag{5.7}$$

The total "likelihood" of a particular value is given by:

$$LH(|z_0|) = 2 \cdot u(|z_0|) = Erf\left(\frac{|z_0|}{\sqrt{2}}, \infty\right) \tag{5.8}$$

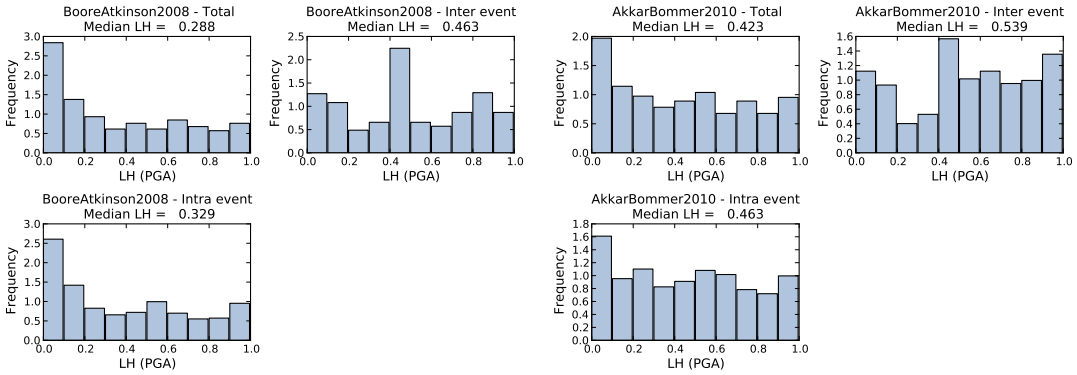
As indicated in SCHERBAUM ETAL 2004 the the LH value should reach a value of 1 for $|z_0| = 0$ (i.e. the mean of the distribution), and should tend to zero for observations further away from the mean. Therefore, if the model assumptions are matched exactly by a set of observations then the $LH(|z_0|)$ values should be distributed evenly between 0 and 1, with a median value of 0.5.

The GMPE-SMTK can produce histograms of the LH value for the ground motion observations using the `res.Likelihood` and `rspl.LikelihoodPlot` tools. The `res.Likeihood` tool operates in the same manner as the `res.Residuals` tool; therefore to run the likelihood analysis one follows the same approach:

```
# Set-up the likelihood calculator
lh1 = res.Likelihood(gmpe_list, imts)
# Run the residual calculator
lh1.get_residuals(sm_database)
```

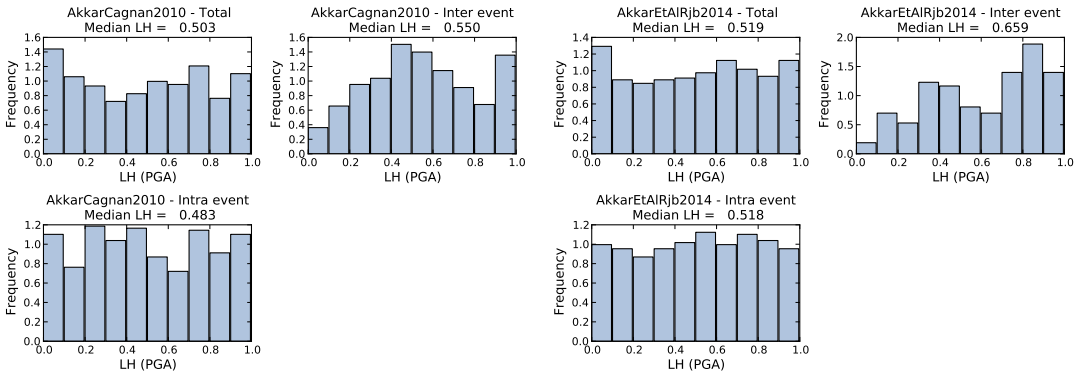
Histograms of the LH values (and their median estimates) for the four GMPEs for PGA are shown in Figure 5.3, and for only Akkar et al. (2014) for $Sa(1.0s)$ in Figure 5.4 . These can be generated with the following command:

```
# Boore & Atkinson (2008)
rspl.LikelihoodPlot(lh1, "BooreAtkinson2008", "PGA",
                    filename="path/to/image.png",
                    filetype="png")
# Akkar & Bommer (2010)
rspl.LikelihoodPlot(lh1, "AkkarBommer2010", "PGA")
# Akkar & Cagnan (2010)
rspl.LikelihoodPlot(lh1, "AkkarCagnan2010", "PGA")
# Akker et al. (2014)
rspl.LikelihoodPlot(lh1, "AkkarEtAlRjb2014", "PGA")
# Akker et al. (2014) - Sa (1.0 s)
rspl.LikelihoodPlot(lh1, "AkkarEtAlRjb2014", "Sa(1.0)")
```



(a) *citeBooreAtkinson2008 model*

(b) *citeAkkarBommer2010 model*



(c) *citeAkkarCagnan2010 model*

(d) *citeAkkarEtAl2014 model*

Figure 5.3 – *LH distribution for the observed PGA set*

In general the trends seen in Figure 5.1 are visible in the *LH* histograms in the manner described in SCHERBAUM ETAL 2004. In the Boore and Atkinson, 2008 model intra-event and total residuals showed a greater variance than predicted by the GMPE, a trend that tends to result in *LH* values closer to zero than 1. For the Akkar et al. (2014) GMPE both the total and inter-event residuals are generally evenly distributed between 0 and 1, with a median of close to 0.5, indicating a good agreement between the model and observations. The inter-event residuals show a smaller range than that expected by the model and so the *LH* values are closer to 1.0 (the distances from the mean are smaller than expected).

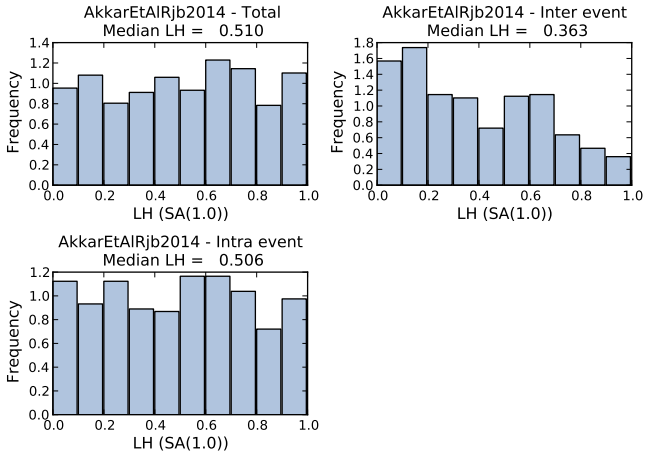


Figure 5.4 – *LH distribution for the observed Sa*$(1.0s)$ *set using the Akkar et al. (2014) GMPE*

5.4 Residual Trends with Predictor Variables

The analysis of the residuals shown in sections 5.2 and 5.3 provide useful insights into the overall fit of a GMPE to a set of observations. They do not necessarily provide a clear indication as to which elements of the model contribute to the misfit. Or, to frame it differently, they do not necessarily indicate where the GMPEs may be biased with respect to the predictor variables of the observations. To analysis this it is necessary to understand if, and where, the GMPE residuals show clear trends with respect to the predictor variables. These trends can indicate those elements of the source/site scaling and attenuation process for which the GMPE is biased.

The GMPE-SMTK offers several methods to visualise the trends of the GMPE residuals with the following predictor variables:

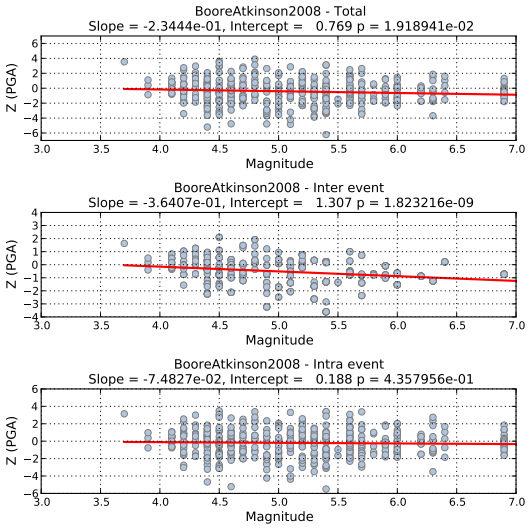
1. Magnitude
2. Distance
3. V_{S30}
4. Hypocentral Depth

Trends with Magnitude

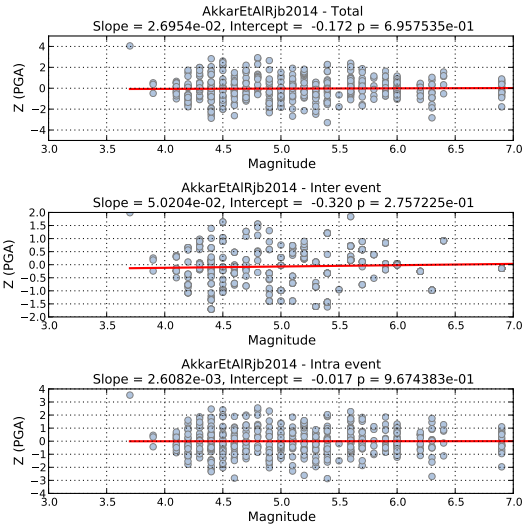
To view the trends with magnitude for a set of ground motion residuals the `rspl` toolset contains the function `ResidualWithMagnitude`. Figure 5.5 shows the residual trends in magnitude for the Boore and Atkinson, 2008 and Akkar et al. (2014) GMPEs for PGA and *Sa*$(1.0s)$. These figures were generated with the following commands:

```
# Boore & Atkinson (2008) - PGA
rspl.ResidualWithMagnitude(resid1, # Residuals
                           "BooreAtkinson2008", # GMPE
                           "PGA", # Intensity Measure
                           filename="path/to/image.png",
                           filetype="png")
# Akkar et al. (2014) - PGA
rspl.ResidualWithMagnitude(resid1, "AkkarEtAlRjb2014", "PGA")
# Boore & Atkinson (2008) - Sa (1.0)
rspl.ResidualWithMagnitude(resid1, "BooreAtkinson2008",
                           "Sa(1.0)")
# Akkar et al. (2014) - Sa (1.0)
rspl.ResidualWithMagnitude(resid1, "AkkarEtAlRjb2014",
                           "Sa(1.0)")
```

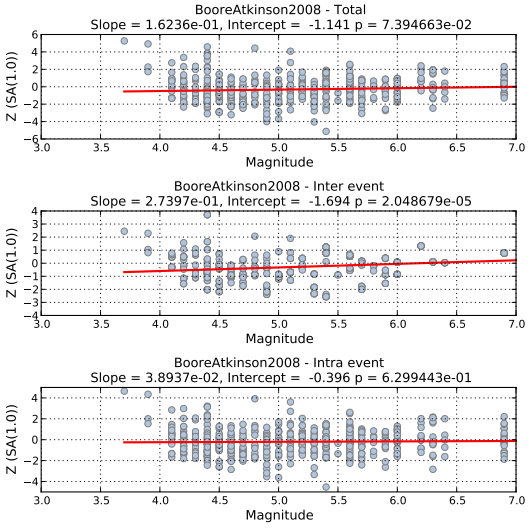
As shown in Figure 5.5, and in subsequent figures, a linear model is fit to the observed residual trends. The significance of the trend is measured via the "p-value", reported in the figure header. This indicates the probability observing the measured trend in the residuals assuming a null hypothesis of no trend (i.e. zero gradient). Consequently a smaller "p-value" indicates a more significant trend, assuming (loosely) that statistically significant trends might be those with a "p-value" smaller than 0.1. If the magnitude scaling is largely unbiased with respect to the GMPE it is expected that no significant linear trend can be observed the plot of residuals with magnitude. This is the case for the Akkar et al. (2014) GMPE in which no statistically significant trend is observed with respect to magnitude. For the Boore and Atkinson (2008) model a statistically significant trend is visible in the inter-event residuals. For PGA this trend is negative, suggesting that the GMPE is predicting higher than observed ground motions for the larget magnitudes, whilst for $Sa\,(1.0s)$ the opposite is true, indicating the GMPE is predicting lower than expected ground motions at higher magnitudes.



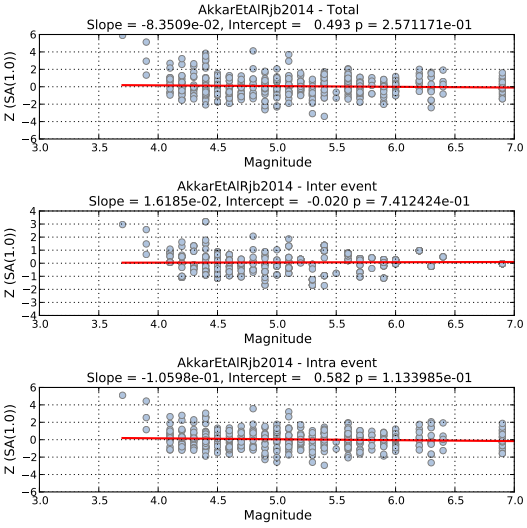
(a) *citeBooreAtkinson2008 model - PGA*



(b) *citeAkkarEtAl2014 model - PGA*



(c) *citeBooreAtkinson2008 model - $Sa\,(1.0s)$*



(d) *citeAkkarEtAl2014 model - $Sa\,(1.0s)$*

Figure 5.5 – *Residual trends with magnitude for the observed PGA and $Sa\,(1.0s)$ records*

Trends with Distance

To observe how well the GMPE compares to the distance attenuation properties in a set of records the `rspl` tools provide the method `ResidualWithDistance`. Plots of residuals with distance for the same GMPEs and intensity measures considered in Figure 5.5 are shown in Figure 5.6. These are generated using the following commands:

```
# Boore & Atkinson (2008)  - PGA
rspl.ResidualWithDistance(resid1,  # Residuals
                          "BooreAtkinson2008",  # GMPE
                          "PGA",   # Intensity Measure
                          distance_type="rjb"
                          plot_type="linear",
                          filename="path/to/image.png",
                          filetype="png")
# Akkar et al. (2014)  - PGA
rspl.ResidualWithDistance(resid1, "AkkarEtAlRjb2014",
                          "PGA", distance_type="rjb",
                          plot_type="linear")
# Boore & Atkinson (2008)  - Sa (1.0)
rspl.ResidualWithDistance(resid1, "BooreAtkinson2008",
                          "Sa(1.0)", distance_type="rjb",
                          plot_type="linear")
# Akkar et al. (2014)  - Sa (1.0)
rspl.ResidualWithDistance(resid1, "AkkarEtAlRjb2014",
                          "Sa(1.0)", distance_type="rjb",
                          plot_type="linear")
```

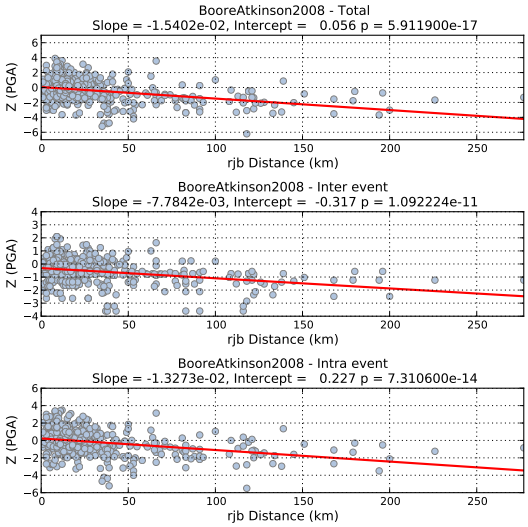
The input arguments are identical to the `ResidualwithMagnitude` tool, with the exception of the `distance_type` and `plot_type` keywords. The `distance_type` determines the distance measure used for visualisation, whilst the `plot_type` keyword indicates whether to plot the distance values on a logarithmic axis ("`log`") or a linear axis ("`linear`"). A logarithmic axis is preferred by default.

In contrast to the residual trends with magnitude, it is evident in Figure 5.6 that statistically significant trends with distance exist for both GMPEs. This can be clearly seen in the intra-event residual, which shows a negative trend in the residuals, a trend that is more pronounced for the Boore & Atkinson (2008) model. This indicates that both models underestimate the degree of attenuation in the ground motions observed in Italy, as both models predict higher ground motion at greater distances than those observed in the strong motion records.

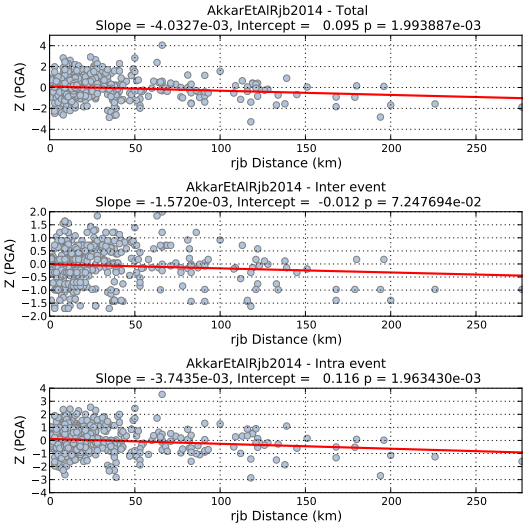
Trends with V_{S30}

The usage of the `rspl` tool to visualise residual trends with V_{S30} (`ResidualWithVs30`) is generally the same as the tools for the magnitude and distance. So for the same GMPEs and intensity measures considered previously, the following commands are used to produce the plot of residuals against V_{S30} for the same record set, shown in Figure 5.7:

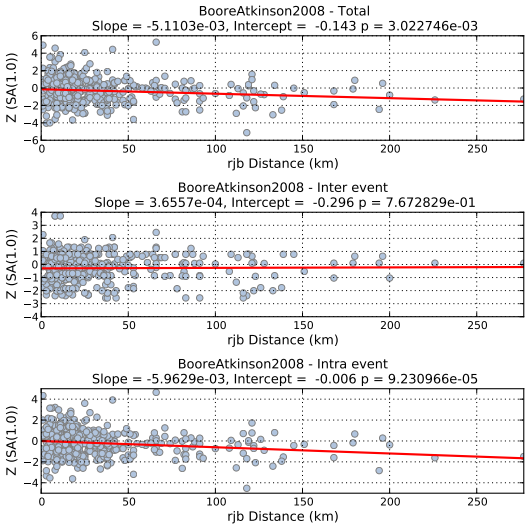
```
# Boore & Atkinson (2008)  - PGA
rspl.ResidualWithVs30(resid1,  # Residuals
                      "BooreAtkinson2008",  # GMPE
                      "PGA",   # Intensity Measure
                      filename="path/to/image.png",
                      filetype="png")
# Akkar et al. (2014)  - PGA
rspl.ResidualWithVs30(resid1, "AkkarEtAlRjb2014", "PGA")
# Boore & Atkinson (2008)  - Sa (1.0)
rspl.ResidualWithVs30(resid1, "BooreAtkinson2008", "Sa(1.0)")
```



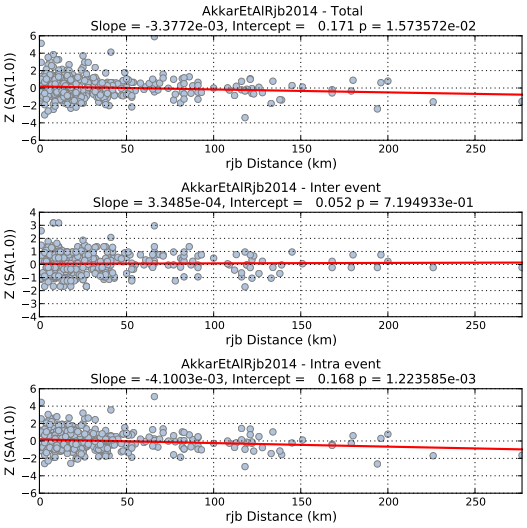
(a) *citeBooreAtkinson2008 model - PGA*



(b) *citeAkkarEtAl2014 model - PGA*



(c) *citeBooreAtkinson2008 model - Sa* $(1.0s)$



(d) *citeAkkarEtAl2014 model - Sa* $(1.0s)$

Figure 5.6 – *Residual trends with Joyner-Boore distance for the observed PGA and Sa* $(1.0s)$ *records*

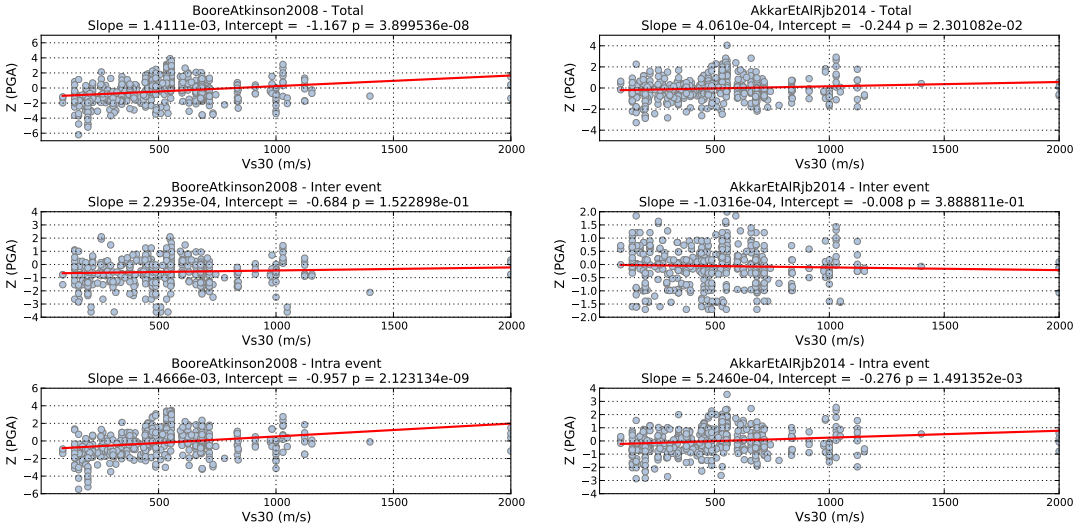
```
11 # Akkar et al. (2014)  - Sa (1.0)
12 rspl.ResidualWithVs30(resid1, "AkkarEtAlRjb2014", "Sa(1.0)")
```

Once again in both the GMPEs and for both intensity measures there are statistically significant trends in the residuals with V_{S30}. In most cases the trends are positive, with a negative intercept, indicating that the GMPEs are predicting higher than expected ground motions on softer soils (i.e. lower V_{s30}) for the selected stations.

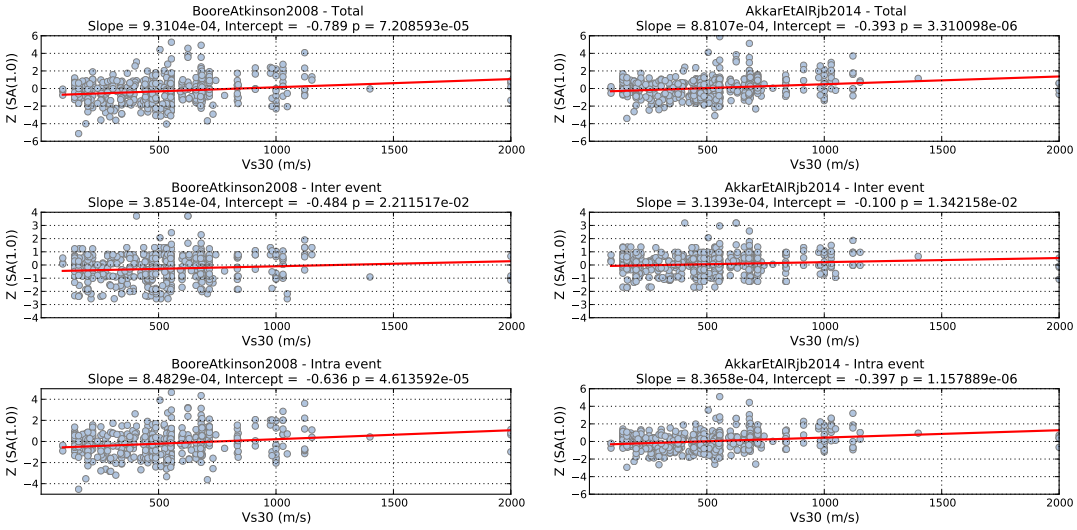
Trends with Hypocentral Depth

Finally, the `rspl` tools contain the method `ResidualWithDepth` to show the trend in the GMPE residuals with *hypocentral* depth. As with the previous residual plotting tools, the commands below will produce plots similar to those shown in Figure 5.8:

```
1 # Boore & Atkinson (2008)  - PGA
2 rspl.ResidualWithDepth(resid1,  # Residuals
3                        "BooreAtkinson2008",  # GMPE
```



(a) *citeAkkarEtAl2014 model - PGA* **(b)** *citeAkkarBommer2010 model - PGA*



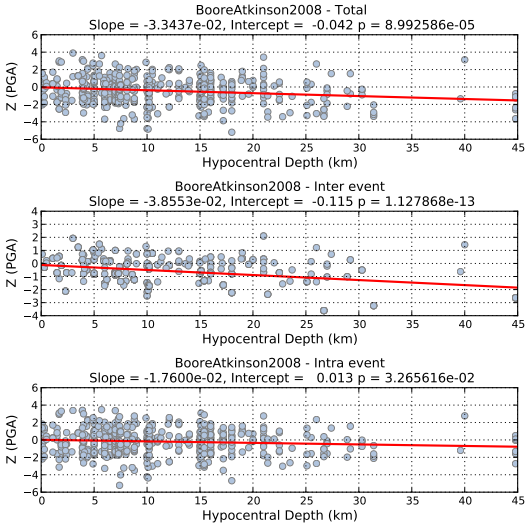
(c) *citeBooreAtkinson2008 model - Sa* $(1.0s)$ **(d)** *citeAkkarEtAl2014 model - Sa* $(1.0s)$

Figure 5.7 – *Residual trends with* V_{S30} *for the observed PGA and Sa* $(1.0s)$ *records*

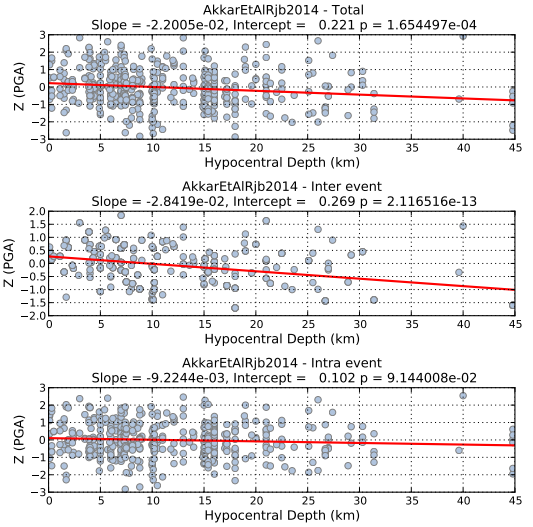
```
                        "PGA",    # Intensity Measure
                        filename="path/to/image.png",
                        filetype="png")
# Akkar et al. (2014)  - PGA
rspl.ResidualWithDepth(resid1, "AkkarEtAlRjb2014", "PGA")
# Boore & Atkinson (2008)  - Sa (1.0)
rspl.ResidualWithDepth(resid1, "BooreAtkinson2008", "Sa(1.0)")
# Akkar et al. (2014)  - Sa (1.0)
rspl.ResidualWithDepth(resid1, "AkkarEtAlRjb2014", "Sa(1.0)")
```

In this case both GMPEs show statistically significant negative trends in the residual with respect to depth. As expected, it is the inter-event residual where these trends are evident, and that control the trends in the total residual. This indicates that the GMPEs are predicting higher ground motions for deeper events than those observed. For the two GMPEs considered, this result is somewhat expected as neither have an explicit dependence on a depth predictor variable;

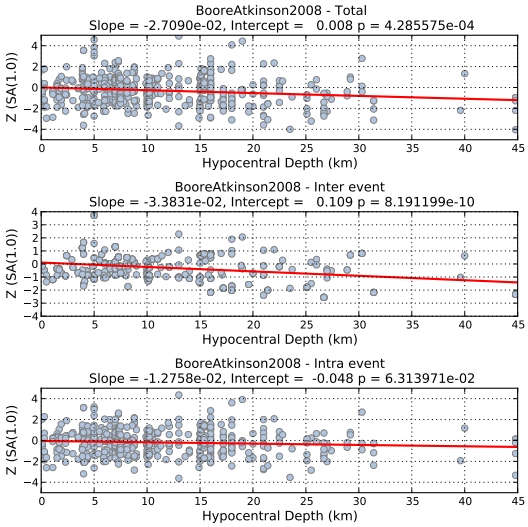
hence hypocentral depth is within the aleatory variability of the model.



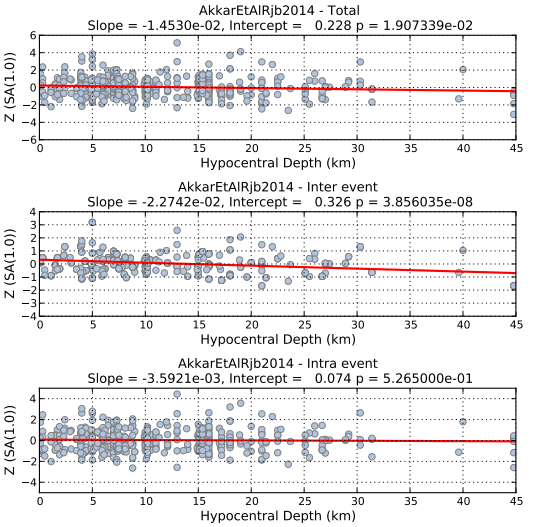
(a) *citeBooreAtkinson2008 model - PGA*



(b) *citeAkkarBommer2010 model - PGA*



(c) *citeBooreAtkinson2008 model - Sa* $(1.0s)$



(d) *citeAkkarEtAl2014 model - Sa* $(1.0s)$

Figure 5.8 – *Residual trends with hypocentral depth for the observed PGA and Sa* $(1.0s)$ *records*

5.5 Log-likelihood Approach

TODO - Add theory description

```
llh_1 = res.LLH(gmpe_list, imt_list)
llh_1.get_residuals(italy_db_clean)
 Run LLH analysis
llh_values, weights = llh_1.get_loglikelihood_values()
```

To view the *LLH* values and weights, it is only necessary to do the following:

```
for gmpe in gmpe_list:
    print "LLH(%s) = %12.6f Weight = %12.6f" %(gmpe,
                                               llh_values[gmpe],
                                               weights[gmpe])
```

which, for the GMPEs and the strong motion database considered throughout this chapter, returns an output similar to that below:

```
LLH(BooreAtkinson2008) =     3.025793  Weight =     0.175937
LLH(AkkarBommer2010) =     2.559079  Weight =     0.243138
LLH(AkkarCagnan2010) =     2.366239  Weight =     0.277909
LLH(AkkarEtAlRjb2014) =     2.241463  Weight =     0.303015
```

5.6 Euclidean Distance Based Ranking

TODO - Add theory description

```
# EDR analysis
edr1 = res.EDR(gmpe_list, imt_list)
edr1.get_residuals(italy_db_clean)
edr_values = edr1.get_edr_values()
```

To view the *EDR* values and weights, it is only necessary to do the following:

```
for gmpe in gmpe_list:
    print "EDR(%s) = %8.4f (sqrt(kappa) = %8.4f)" %(gmpe,
        edr_values[gmpe]["EDR"], edr_values[gmpe]["sqrt Kappa"])
```

```
EDR(BooreAtkinson2008) =   1.2163 (sqrt(kappa) =   1.1840)
EDR(AkkarBommer2010) =   1.0990 (sqrt(kappa) =   1.0609)
EDR(AkkarCagnan2010) =   1.3704 (sqrt(kappa) =   1.1308)
EDR(AkkarEtAlRjb2014) =   1.0956 (sqrt(kappa) =   1.0889)
```

5.7 Site Specific Analysis

TODO

6. Hazard Applications

6.1 Developing GMPE Logic Trees: A "Good Practice" Guide

(TODO) - If possible I would like some of these issues to be discussed at the meeting.

6.2 Conditional Field Simulation

The extraction of the ground motion residuals from a set of observations also opens up the possibility of implementing another type of calculation that is becoming increasingly common in seismic hazard and risk modelling, and that is conditional random field simulation. These are randomly generated fields of ground motion residuals (z) whose values are i) spatially correlated according to a multivariate Gaussian distribution, and ii) conditioned upon observed (known) values of the residual. This type of analysis can be critical in characterising the uncertainty in "ShakeMaps" (i.e. field of ground motion generated from an observed event), which in turn can provide insights into the distribution of expected or, if applied *a posteriori*, observed losses (citeParketal, Crowleyetal2008, Staffordetal2012).

It has been well-established from observations of densely recorded earthquakes, that ground motion residuals are spatially correlated when stations are separated by distances typically on the order of a few tens of kilometres (cite Boore2003, JayaramBaker2008). Given the assumption of lognormality in the GMPEs, it is has also been demonstrated appropriate to model the spatial correlation in ground motion residual between two or more sites using a multivariate Gaussian distribution (cite JayaramBaker2009). The coeffiecient of correlation between two sites separated by a distance h ($\rho(h)$) is shown to decay exponentially with distance such that:

$$\rho(h) = \exp\left(\frac{-a(T)h}{b(T)}\right) \tag{6.1}$$

where $a(T)$ and $b(T)$ control the shape of the decay and are dependent on the period of ground motion under consideration. One such model of spatial correlation, which we shall use in this

example, is that of cite JayaramBaker2009:

$$\rho\left(h\right) = \exp\left(\frac{-3h}{b\left(T\right)}\right) \quad \text{where} \quad b\left(T\right) = \begin{cases} \begin{cases} 8.5 + 17.2T \text{ “Case 1”} & T < 1\ s \\ 40.7 - 15.0T \text{ “Case 2”} & T < 1\ s \end{cases} \\ 22.0 + 3.7T \qquad\qquad T \geq 1\ s \end{cases} \tag{6.2}$$

Consider a set of correlated random variables distributed according to a multivariate Gaussian distribution with a mean of zero and a standard deviation of unity ($\mathbf{X} = [X_1, X_2, \ldots, X_N]$) and with a positive definite covariance matrix given by $\mathbf{C}$. A randomly generated sample of values can be generated via:

$$\mathbf{X} = \mu + \mathbf{L}\mathbf{U} \tag{6.3}$$

where μ is the mean values of the field (in this case a zero-vector $\mathbf{0}$), $\mathbf{U}$ is a sample of independent normally distributed random variates and $\mathbf{L}$ is the lower matrix defined by $\mathbf{L}\mathbf{L}^{\mathbf{T}} = \mathbf{C}$, which is obtained by Cholesky factorisation of the covariance matrix. This assumes that values of $\mathbf{X}$ are unknown *a priori*, as might be the case if considering spatially correlated fields of ground motion residuals for some as yet unrecorded event. The ability to generate spatially correlated fields in this manner is supported directly by OpenQuake.

When considering a recorded event it is possible to partition the ground motion field into those locations for which the GMPE residual is known ($\mathbf{X_{OBS}}$), and those for which it is unknown ($\mathbf{X_{UNK}}$). Given the locations of both the known and unknown values the covariance matrices can be formed where $\mathbf{C_{11}}$ is the covariance of the known residuals, $\mathbf{C_{22}}$ the covariance of the unknown locations and $\mathbf{C_{12}}$ and $\mathbf{C_{21}}$ the covariance matrices considering the distance between the known and unknown locations. In the case of a multivariate standard normal distribution, the distribution of residuals at both locations can be depicted as:

$$\left[\begin{array}{c} \mathbf{X_{OBS}} \\ \hline \mathbf{X_{UNK}} \end{array}\right] \sim N_M \left(\left[\begin{array}{c} \mathbf{0} \\ \hline \mathbf{0} \end{array}\right], \left[\begin{array}{c|c} \mathbf{C_{11}} & \mathbf{C_{12}} \\ \hline \mathbf{C_{21}} & \mathbf{C_{22}} \end{array}\right] \right) \tag{6.4}$$

which is arranged to give the distribution of residuals at the unknown locations, conditional upon the known residuals:

$$[\mathbf{X_{UNK}}|\mathbf{X_{OBS}}] \sim N_M \left(\left[\mathbf{C_{21}}\mathbf{C_{11}^{-1}}\mathbf{X_{OBS}}\right], \left[\mathbf{C_{22}} - \mathbf{C_{21}}\mathbf{C_{11}^{-1}}\mathbf{C_{12}}\right] \right) \tag{6.5}$$

From this model it is then possible to simulate random fields of GMPE residuals conditioned upon observed residuals. This functionality is not directly supported in OpenQuake, as it requires a mean by which the observed residuals can be characterised using the desired GMPE. As has been shown in Chapter 5, however, the GMPE-SMTK does exactly this. Therefore it becomes possible to use the GMPE-SMTK, in conjunction with OpenQuake, to produce multiple fields of spatially correlated ground motions that are conditional upon a set of known observations. For this purpose we include a set of tools in a module called `smtk.hazard.conditional_simulation`, which we shall import as, and subsequently refer to as, `csim`

```
1 | import smtk.hazard.conditional_simulation as csim
```

6.2.1 Example of Conditional Simulation: L'Aquila

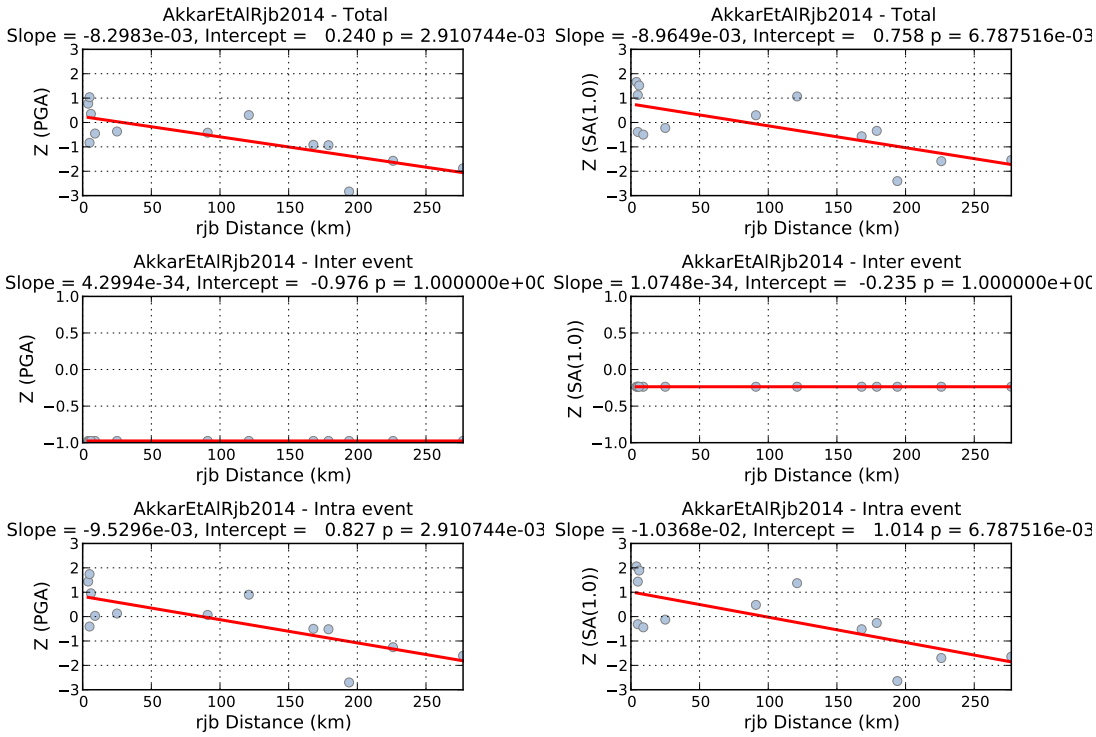
In the following example we consider a set of 15 records taken from the L'Aquila earthquake. In the current example these records have been put into a GMPE-SMTK database on the path `/data/LAquila_Database/`, following the approach described in section 2.4. The first step in this process is to load the database and calculate the ground motion residuals:

```
import cPickle
from smtk.residuals.gmpe_residuals import Residuals
# Load the database
import cPickle
db1 = cPickle.load(open("./data/LAquila_Database/metadatafile.pkl",
                        "r"))
```

In this example we shall consider only the citeAkkaretal2014 GMPE, and just two intensity measures: PGA and $Sa\,(1.0s)$:

```
gmpe_list = ["AkkarEtAlRjb2014"]
imts = ["PGA", "SA(0.2)", "SA(1.0)"]
# Calculate the residuals
resid1 = Residuals(gmpe_list, imts)
resid1.get_residuals(db1)
```

We may be interested to observe the distance trend in these residuals. This is generated using the tools described in chapter 5, and can be seen in Figure 6.1.



(a) *PGA* **(b)** $Sa\,(1.0s)$

Figure 6.1 – *Residual trends with distance for the L'Aquila ground motion residuals*

Here we see that for both PGA and $Sa\,(1.0s)$ the inter-event residual is moderately negative (-0.976 for PGA and -0.235 for $Sa\,(1.0s)$), and we observe a clear trend that the ground motions from this earthquake attenuated to a greater extent than that predicted by the GMPE.

In the present case we have a model of the finite extent of the earthquake rupture. We put this rupture into an OpenQuake xml format[1], as shown below, and save it in the file `laquila_rupture.xml`.

```
<?xml version='1.0' encoding='utf-8'?>
<nrml xmlns:gml="http://www.opengis.net/gml"
      xmlns="http://openquake.org/xmlns/nrml/0.4">
    <simpleFaultRupture>
        <magnitude>6.3</magnitude>
        <rake>-98.0</rake>
        <hypocenter lon="13.38" lat="42.3476" depth="9.3"/>
        <simpleFaultGeometry>
                <gml:LineString>
                    <gml:posList>
                        13.3976 42.4371
                        13.5571 42.3012
                    </gml:posList>
                </gml:LineString>
            <dip>50.0</dip>
            <upperSeismoDepth>0.8</upperSeismoDepth>
            <lowerSeismoDepth>12.69</lowerSeismoDepth>
        </simpleFaultGeometry>
    </simpleFaultRupture>
</nrml>
```

With the use of the OpenQuake hazardlibrary and Python's `Matplotlib` plotting tools we can visualise the spatial trend in the residuals. Firstly we load in the rupture. The `csim` module contains the method `build_rupture_from_file`, which will load in the rupture:

```
rupture_file = "laquila_rupture.xml"
rupture = csim.build_rupture_from_file(rupture_file)
```

For plotting it is convenient to define the outline of the surface projection of the rupture as a simple polygon:

```
# Import the OpenQuake polygon class
from openquake.hazardlib.geo.polygon import Polygon
rupture_outline = Polygon([rupture.surface.top_left,
                           rupture.surface.top_right,
                           rupture.surface.bottom_right,
                           rupture.surface.bottom_left,
                           rupture.surface.top_left])
```

The next step is to take the database and return its locations as an instance of the class `openquake.hazardlib.site.SiteCollection`. This can be done using the method within the database class (`GroundMotionDatabase`) called `get_site_collection`

```
observed_sites = db1.get_site_collection()
```

The plot of the spatial distribution of residuals of PGA with respect to the rupture location is shown in Figure 6.2. This can be generated using the Python code below:

```
# Import the Matplotlib library
import matplotlib.pyplot as plt
```

[1] More information on the OpenQuake scenario rupture format can be found in citeCrowley2013

```
# Extract the PGA residuals
pga_residuals = resid1.residuals["AkkarEtAlRjb2014"]["PGA"]["Intra event"]
plt.figure(figsize=(10,8))
# Plot the rupture
plt.plot(rupture_outline.lons, rupture_outline.lats, "r-")
# Plot the reisduals
plt.scatter(observed_sites.lons, observed_sites.lats,
            s=40,
            c=pga_residuals,
            norm=Normalize(vmin=-3.0, vmax=3.0))
plt.title("PGA Observed Intra-event Residual", fontsize=16)
plt.colorbar()
plt.xlabel("Longitude", fontsize=14)
plt.ylabel("Latitude", fontsize=14)
```

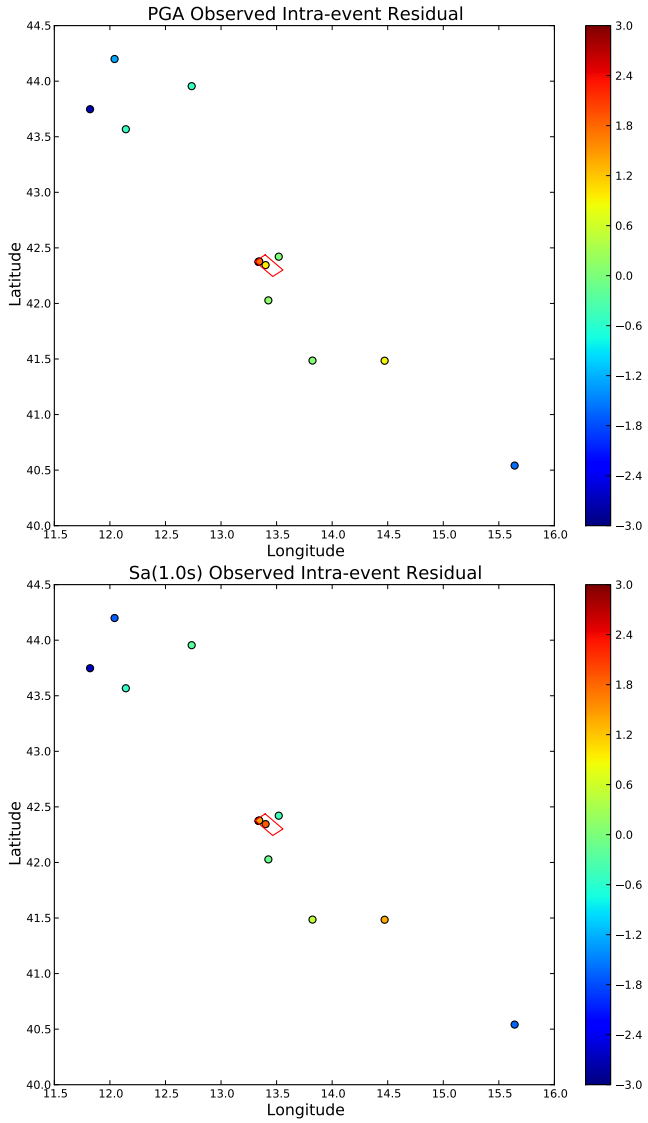


Figure 6.2 – *Distribution of intra-event residuals for the L'Aquila ground motion records. The surface projection of the L'Aquila rupture is shown in red.*

Figure 6.2 helps understand the residual distribution further as we observed that for both intensity measures there is a small cluster of strongly positive residuals in the very near-field of

the fault (due to directivity effects).

If we want to generate a spatially correlated random field of observations around the L'Aquila area, we can use the observed GMPE residuals and the spatial correlation model in equation 6.2 to generate a realisation of a field of possible residuals. In the present example we have created a regularly spaced grid of "target" (i.e. unknown sites) with known soil properties. We assume this has been created as an instance of the `openquake.hazardlib.site.SiteCollection` class, and we call this variable `unknown_sites`. To generate a 10 random fields of spatially correlated residuals conditioned upon a set of observations we can use the `csim` method `conditional_simulation` as follows:

```
# Generate fields of conditioned on the PGA residuals
output_resid_pga = csim.conditional_simulation(observed_sites,
                                               pga_residuals,
                                               unknown_sites,
                                               "PGA",
                                               nsim=10)
# Generate fields of conditioned on the Sa (1.0) residuals
output_resid_sa1 = csim.conditional_simulation(observed_sites,
                                               pga_residuals,
                                               unknown_sites,
                                               "SA(1.0)",
                                               nsim=10)
```

The output will be an array of shape $N_{SITES} \times N_{SIMULATIONS}$, where N_{SITES} is the number of unknown sites and $N_{SIMULATIONS}$ the number of desired random fields (10 in this example). For a field located close to the rupture, the spatially correlated residuals are shown for both intensity meausres in Figures 6.3 and Figures 6.4 respectively.

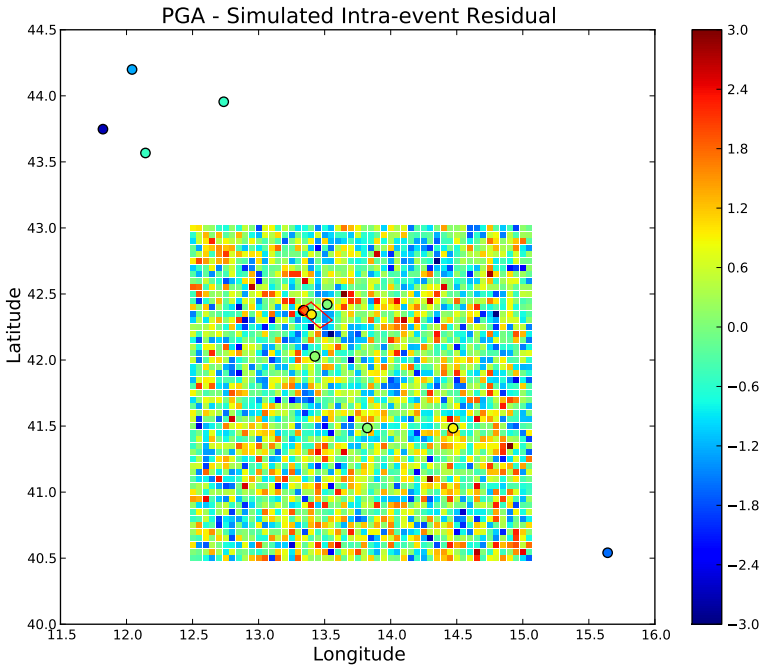


Figure 6.3 – *Random field of PGA ground motion residuals conditioned upon observed residuals (circles) in the L'Aquila region*

The spacing of this particular field is approximately 5 km. The typical correlation length (i.e. the distance at which $\rho(h)$ drops below 0.05) of PGA is on the order of approaximately 10 to

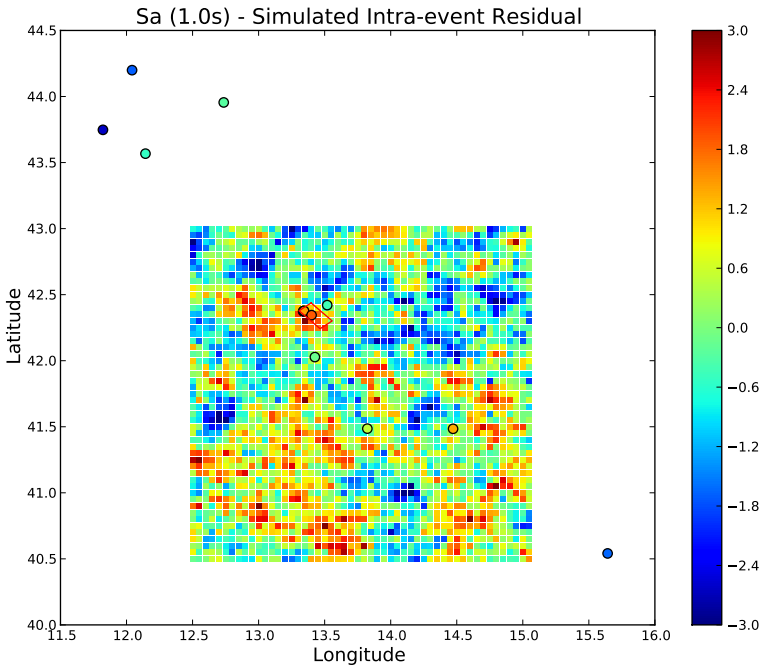


Figure 6.4 – *Random field of* $Sa\,(1.0s)$ *ground motion residuals conditioned upon observed residuals (circles) in the L'Aquila region*

15 km. Given the exponential model it is evident that for PGA the spatial correlation is not so significant at this resolution. For $Sa\,(1.0s)$ the correlation length may be on the order of 30 to 40 km; hence both the spatial correlation and the effects of the conditioning are visible clearly in Figure 6.4. Particular attention should be paid to the area of strong positive residuals in the near-field of the rupture.

The `csim` tools contain a general method to implement much of the process outlined previously. This method is called `get_conditional_gmfs` and its usage is illustrated below:

```
gmfs = csim.get_conditional_gmfs(db1,
                                 rupture,
                                 sites=unknown_sites,
                                 gsims=["AkkarEtAlRjb2014"],
                                 imts=["PGA", "SA(1.0)"],
                                 number_simulations=5,
                                 truncation_level=3.0)
```

The above command requires as input:

1. The database of observed ground motions for the rupture
2. The loaded rupture model
3. The target sites as an instance of the `SiteCollection` class
4. The list of GMPEs
5. The list of intensity measures
6. The number of simulations
7. The number of standard deviations at which to truncate the residuals.

The following code snippet can be used to generate the full simulated ground motion field, similar to the examples shown in Figures 6.5 and 6.6:

```
# Get one of the PGA fields from the Akkar et al GMPe
pga_field = gmfs["AkkarEtAlRjb2014"]["PGA"][:, 0]
```

```
# Setup the figure
plt.figure(figsize=(10,8))
plt.plot(rupture_outline.lons, rupture_outline.lats, "r-")
# Plot the fields
plt.scatter(unknown_sites.lons, unknown_sites.lats,
            s=50,
            c=pga_field,
            marker="s",
            edgecolor="None",
            norm=LogNorm(vmin=0.001, vmax=1))
plt.xlim(12.5, 15.0)
plt.ylim(40.5, 43.0)
plt.title("PGA (g) - Conditional Random Field", fontsize=18)
plt.colorbar()
plt.xlabel("Longitude", fontsize=14)
plt.ylabel("Latitude", fontsize=14)
```

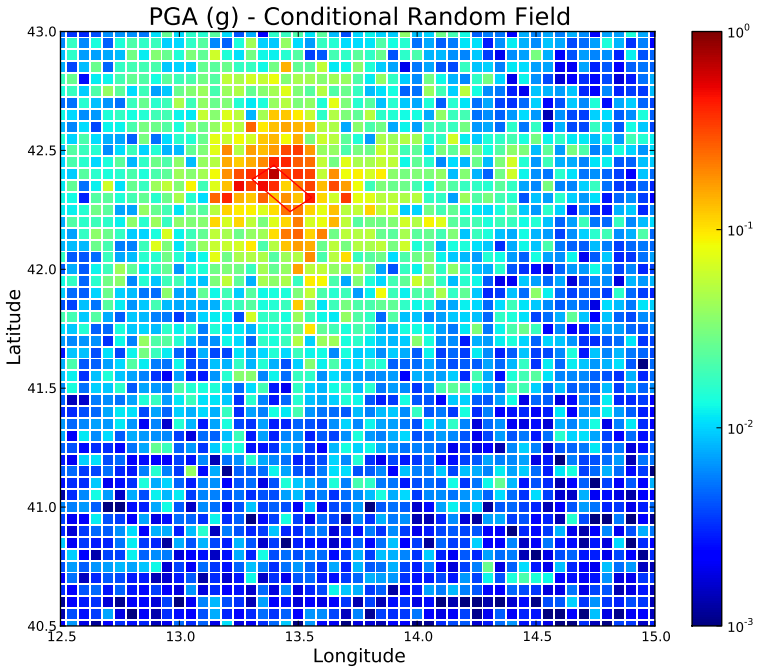


Figure 6.5 – *PGA ground motion field conditioned upon the ground motion residuals conditioned upon observed ground motions from the L'Aquila (2009) earthquake*

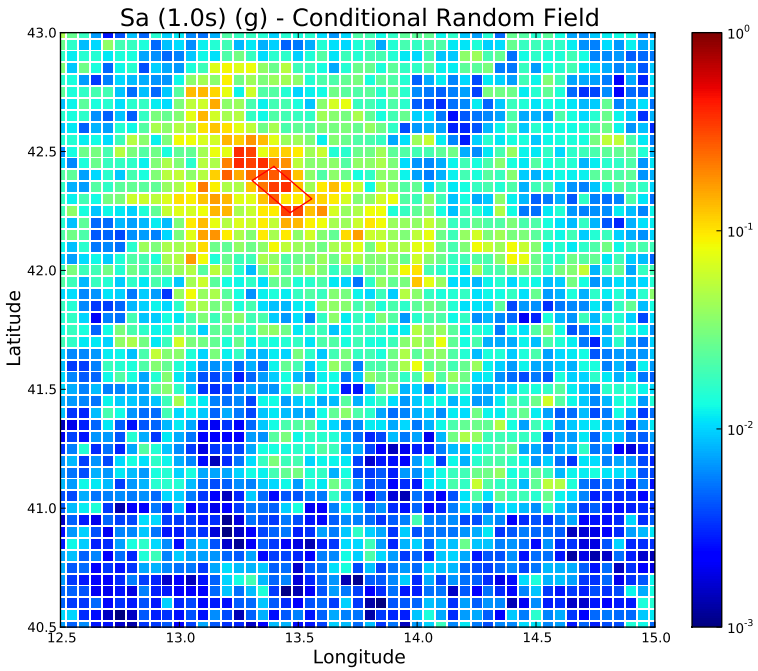


Figure 6.6 – *Sa* $(1.0s)$ *ground motion field conditioned upon the ground motion residuals conditioned upon observed ground motions from the L'Aquila (2009) earthquake*

Bibliography

Books

Articles

Boore, D. M. and G. M. Atkinson (Feb. 2008). “Ground-Motion Prediction Equations for the Average Horizontal Component of PGA, PGV, and 5%-Damped PSA at Spectral Periods between 0.01 s and 10.0 s”. In: *Earthquake Spectra* 24.1, pages 99–138 (cited on pages 52, 54, 55).

Wells, D. L. and K. J. Coppersmith (1994). “New Empirical Relationships among Magnitude, Rupture Length, Rupture Width, Rupture Area, and Surface Displacement”. In: *Bulletin of the Seismological Society of America* 84.4, pages 974–1002 (cited on page 40).

Other Sources