

“OpenQuake: Calculate, share, explore”

Ground Motion Prediction Equation (GMPE) and Strong Motion Modeller's Toolkit - User Guide

Copyright © 2014 GEM Foundation

PUBLISHED BY GEM FOUNDATION

GLOBALQUAKEMODEL.ORG/OPENQUAKE

Disclaimer

The “Ground Motion Prediction Equation (GMPE) and Strong Motion Modeller’s 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.

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

This Book is distributed under the Creative Common License Attribution-NonCommercial-NoDerivs 3.0 Unported (CC BY-NC-ND 3.0) (see link below). You can download this Book and share it with others as long as you provide proper credit, but you cannot change it in any way or use it commercially.

First printing, June 2014

needs to be up-
or removed

Contents

1	Introduction	5
1.1	A Ground Motion Prediction Equation (GMPE) Toolkit	5
1.2	Learning Objectives and Development Philosophy	5
1.3	Installation and Getting Started	5
2	The Ground Motion Database	7
2.1	Strong Motion Observations	7
2.2	Current Data Model	7
2.2.1	Source Characteristics	9
2.2.2	Site Characteristics	10
2.2.3	Waveform Characteristic	11
2.2.4	Geometry Characterisations	11
2.3	Missing Metadata!	11
2.4	The HDF5 Strong Motion Database	11
2.4.1	Loading Data to the Database	12
2.5	Selecting Subsets of Records from the Database	14
3	Response Spectra and Intensity Measures	17
3.1	Ground Motion Waveforms	17
3.2	Scalar Intensity Measures	19
3.2.1	Peak Measures	19
3.2.2	Duration	19
3.2.3	Other Scalar Measures	19
3.3	Response Spectra	19
3.4	Combining Horizontal Components of Motion	22

3.5	Rotation of the Horizontal Records	23
3.6	Adding Horizontal Components to the Database	24
4	Comparing GMPEs: Trellis Plotting	27
4.1	Trellis Plots: Advantages & Limitations	27
4.2	Trellis Plotting: The (Not So) Simple Way	28
4.2.1	Comparing the Scaling with Magnitude	28
4.2.2	Comparing the Scaling with Distance	30
4.2.3	Comparing the Scaling of the Response Spectrum with Magnitude and Distance	31
4.3	Trellis Plotting: The Simple Way	33
4.3.1	Configuring the Sites	36
4.3.2	Creating the Trellis Points	38
5	Comparing Models and Observations	45
5.1	GMPE Residuals	45
5.1.1	Definition of the GMPE Residuals	45
5.1.2	Residual analysis in the GMPE-SMTK	46
5.2	Basic Trends in Residuals	47
5.3	The Likelihood Model (TOCITE Scherbaum et al., 2004)	49
5.4	Residual Trends with Predictor Variables	51
5.5	Log-likelihood Approach	56
5.6	Euclidean Distance Based Ranking	57
5.7	Site Specific Analysis	57
6	Hazard Applications	59
6.1	Developing GMPE Logic Trees: A “Good Practice” Guide	59
6.2	Conditional Field Simulation	59
	Bibliography	61
	Books	61
	Articles	61
	Reports	61

1. Introduction

- 1.1 A Ground Motion Prediction Equation (GMPE) Toolkit
- 1.2 Learning Objectives and Development Philosophy
- 1.3 Installation and Getting Started

Strong Motion Observations

Current Data Model

- Source Characteristics
- Site Characteristics
- Waveform Characteristic
- Geometry Characterisations

Missing Metadata!

The HDF5 Strong Motion Database

- Loading Data to the Database

Selecting Subsets of Records from the Database

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 OpenQuake's GMPE's¹. 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` function 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**

¹Their 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.

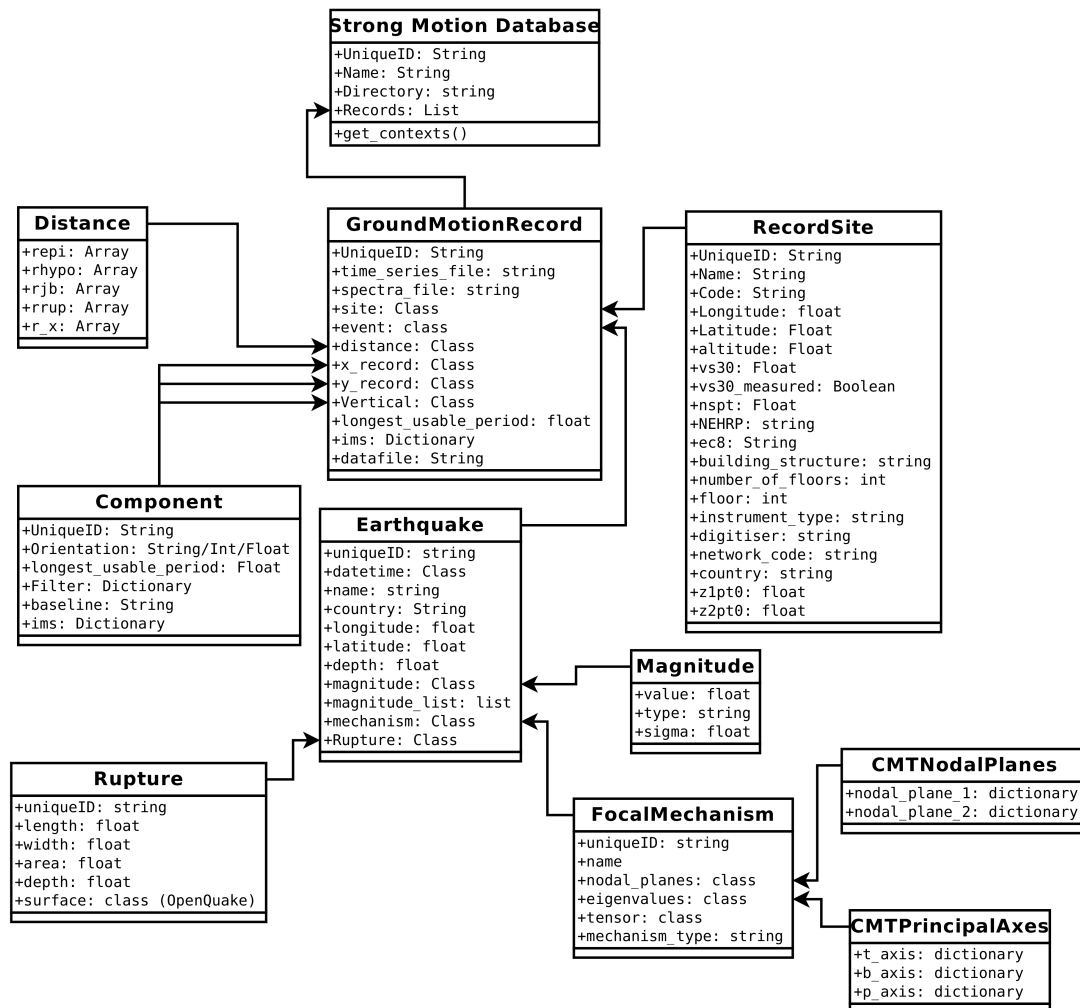


Figure 2.1 – Core Architecture of the Strong Motion Data Model

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:

```
1 | from datetime import datetime
2 | # The following event occurs at 2010/6/1 20:36:27.4
3 | 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:

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

```

5 | nps.nodal_plane_1 = {"strike": 30., "dip": 90., "rake" = 0.}
6 | 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:

```

1 | # Mechanism with the following planes
2 | # T-axis = Eigenvalue 1.5E24, Plunge = 8, Azimuth = 203
3 | # B-axis = Eigenvalue -0.11E24, Plunge = 77, Azimuth = 332
4 | # P-axis = Eigenvalue -1.393, Plunge = 10, Azimuth = 111
5 | from smtk.sm_database import GMCTPrincipalAxes
6 | pax1 = GMCTPrincipalAxes()
7 | pax1.t_axis = {"eigenvalue": 1.5E24,
8 |               "azimuth": 203.,
9 |               "plunge": 8.}
10 | pax1.b_axis = {"eigenvalue": -0.11E24,
11 |               "azimuth": 332.,
12 |               "plunge": 77.}
13 | pax1.p_axis = {"eigenvalue": -1.393E24,
14 |               "azimuth": 111.,
15 |               "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 (km²)
- **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 relevant 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 information 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:

```
1 | filter_data = {"Type": "Butterworth",
2 |               "Order": 2,
3 |               "Passes": 2,
4 |               "Low-Cut": 0.05,
5 |               "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) `SMTTimeSeriesReader` 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:

```

1  # Import the readers for database XXXXX
2  from smtk.parsers.XXX_database_parser import (XXXMetadataReader,
3                                               XXXTimeSeriesReader,
4                                               XXXSpectraReader)
5  # Import the database builder
6  from smtk.sm_database_builder import SMDatabaseBuilder,
7
8  # Path to the directory containing the input files
9  input_fileset = "./path/to/strong/motion/folder"
10 # Path to the directory intended for the database
11 output_database = "./path/to/output/database/folder"
12
13 # Create an instance of the database builder
14 db_builder = SMDatabaseBuilder(XXXMetadataReader,
15                               output_database)
16
17 # Create the metadata database
18 db_builder.build_database("UNIQUEID", # Database ID
19                          "DATABASE NAME", # Name
20                          input_fileset)
21

```

³<http://www.hdfgroup.org/products/java/hdfview/>

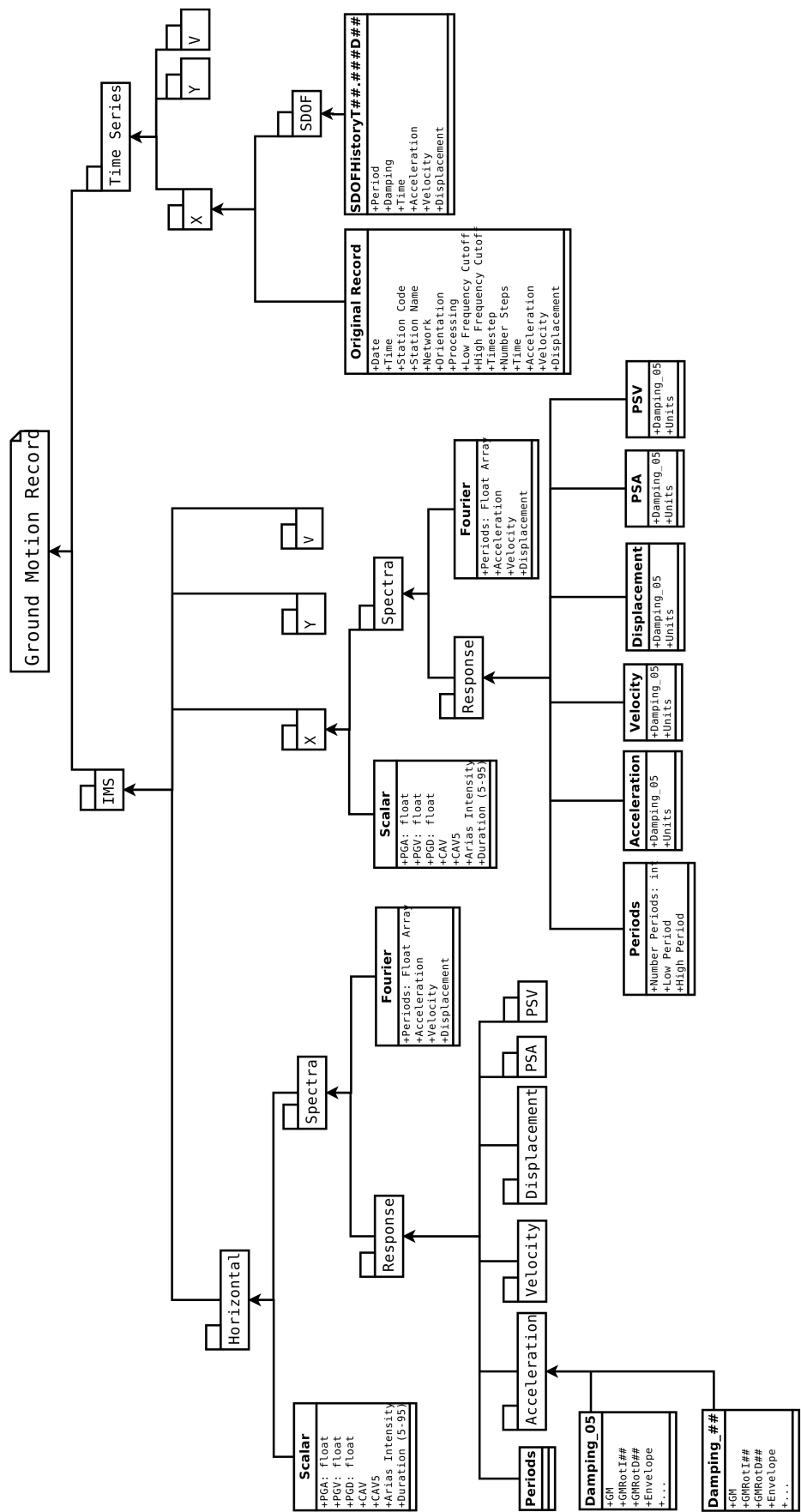


Figure 2.2 – Architecture of the hdf5 representation of a strong motion record

```

22 # Parse the strong motion records
23 db_builder.parse_records(XXXTimeSeriesReader,
24                          XXXSpectraReader)
25
26 # (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:

```

1 # Import the cPickle module
2 import cPickle
3 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:

```

1 # Import the selector
2 from smtk.strong_motion_selector import SMRecordSelector
3 # Create an instance of the class for a specific database
4 selector = SMRecordSelector(db1)
5
6 # Select the 3rd, 6th and 17th record in the database
7 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:

```

1 # Select records between 1990/01/01 and 2010/12/31
2 db2 = selector.select_within_time(datetime(1990, 1, 1),
3                                   datetime(2010, 12, 31),
4                                   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:

```
1 db2 = selector.select_by_station_country("Italy",
2                                         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:

```
1 db2 = selector.select_by_site_attribute("building_structure",
2                                       "Free-Field",
3                                       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 `Vs30` 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:

```
1 from openquake.hazardlib.geo.point import Point
2 location = Point(30.0, 40.0)
3 db2 = selector.select_stations_within_distance(location,
4                                               200.0,
5                                               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:

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



```

2 | from openquake.hazardlib.geo.polygon import Polygon
3 |
4 | region = Polygon([Point(30.0, 40.0),
5 |                  Point(40.0, 40.0),
6 |                  Point(40.0, 20.0),
7 |                  Point(30.0, 20.0)])
8 | db2 = selector.select_stations_within_region(region,
9 |                                              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.

```

1 | db2 = selector.select_within_distance_range(
2 |     "rjb", 5.0, 50.0,
3 |     alternative=("repi", 10.0, 70.0),
4 |     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:

```

1 | db2 = selector.select_mechanism_type("strike-slip",
2 |                                     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)

[illegible]

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:

```
1 rsp.plot_time_series(accel_x,
2                       time_step,
3                       velocity=vel_x,
4                       displacement=disp_x,
5                       units="cm/s/s",
6                       filename="path/to/output/image.eps",
7                       filetype="eps",
8                       dpi=300,
9                       linewidth=1.5)
```

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

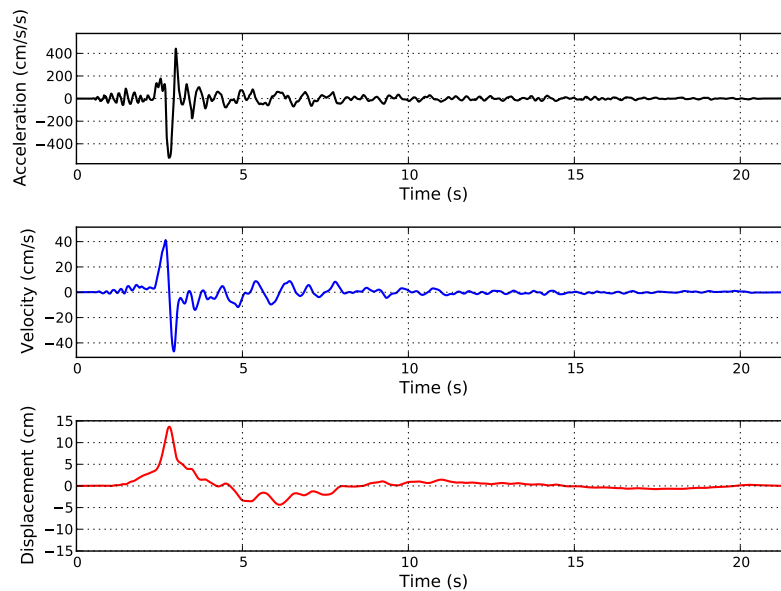


Figure 3.1 – Example time series for a single record: acceleration (top), velocity (middle) and displacement (bottom)

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:

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

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:

```
1 | pga, pgv, pgd, vel_x, disp_x = get_peak_measures(time_step,
2 |                                     accel_x,
3 |                                     get_vel=True,
4 |                                     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 of single-degree-of-freedom (SDOF) oscillators with different natural periods (T), and damping ratio ξ . The pseudospectral acceleration at a given period $S_a(T, \xi)$, and its corresponding pseudospectral velocity and displacements, $S_v(T, \xi)$ and $S_d(T, \xi)$ 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:

```
1 | # Define the periods for calculating spectral acceleration
2 | periods = np.array([0.01, 0.02, 0.03, 0.04, 0.05, 0.075, 0.1,
3 |                    0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17,
4 |                    0.18, 0.19, 0.20, 0.22, 0.24, 0.26, 0.28,
5 |                    0.30, 0.32, 0.34, 0.36, 0.38, 0.40, 0.42,
6 |                    0.44, 0.46, 0.48, 0.50, 0.55, 0.6, 0.65,
7 |                    0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00,
8 |                    1.10, 1.20, 1.30, 1.40, 1.50, 1.60, 1.70,
9 |                    1.80, 1.90, 2.00, 2.20, 2.40, 2.60, 2.80,
10 |                   3.00, 3.20, 3.40, 3.60, 3.80, 4.00, 4.20,
11 |                   4.40, 4.60, 4.80, 5.00, 5.50, 6.00, 6.50,
12 |                   7.00, 7.50, 8.00, 8.50, 9.00, 9.50, 10.0])
```

```

13 # Call the Newmark-Beta methodology
14 newmark_beta = rsp.NewmarkBeta(acceleration,
15                                time_step,
16                                periods,
17                                damping=0.05,
18                                units="cm/s/s")
19
20 spectra, time_series, acc, vel, dis = newmark_beta.evaluate()
21
22 # Call the Nigam & Jennings (1969) method
23 nigam_jennings = rsp.NigamJennings(acceleration,
24                                    time_step,
25                                    periods,
26                                    damping=0.05,
27                                    units="cm/s/s")
28
29 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:

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

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

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

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:

```
1 spectra, time_series, acc, vel, dis = ims.get_response_spectrum(
2     accel_x,
3     time_step,
4     periods,
5     damping=0.05,
6     units="cm/s/s",
7     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:

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

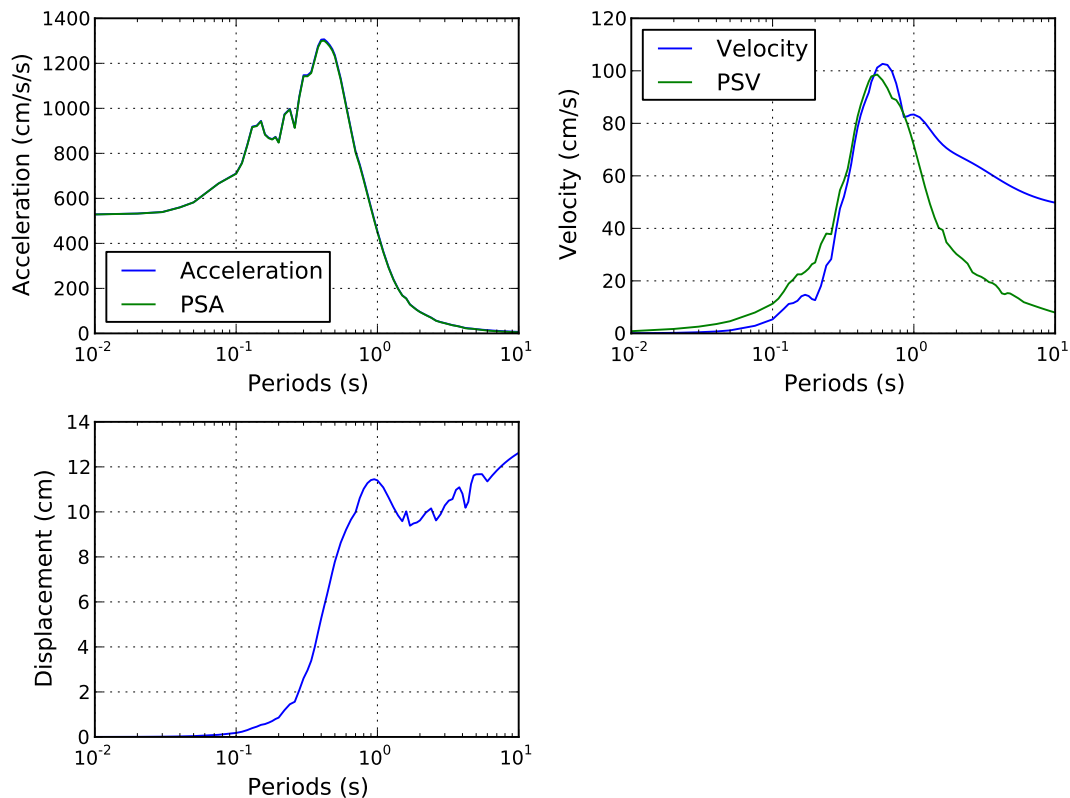


Figure 3.2 – Full response spectra of the waveform shown in Figure 3.1, calculated using the Newmark- β method

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

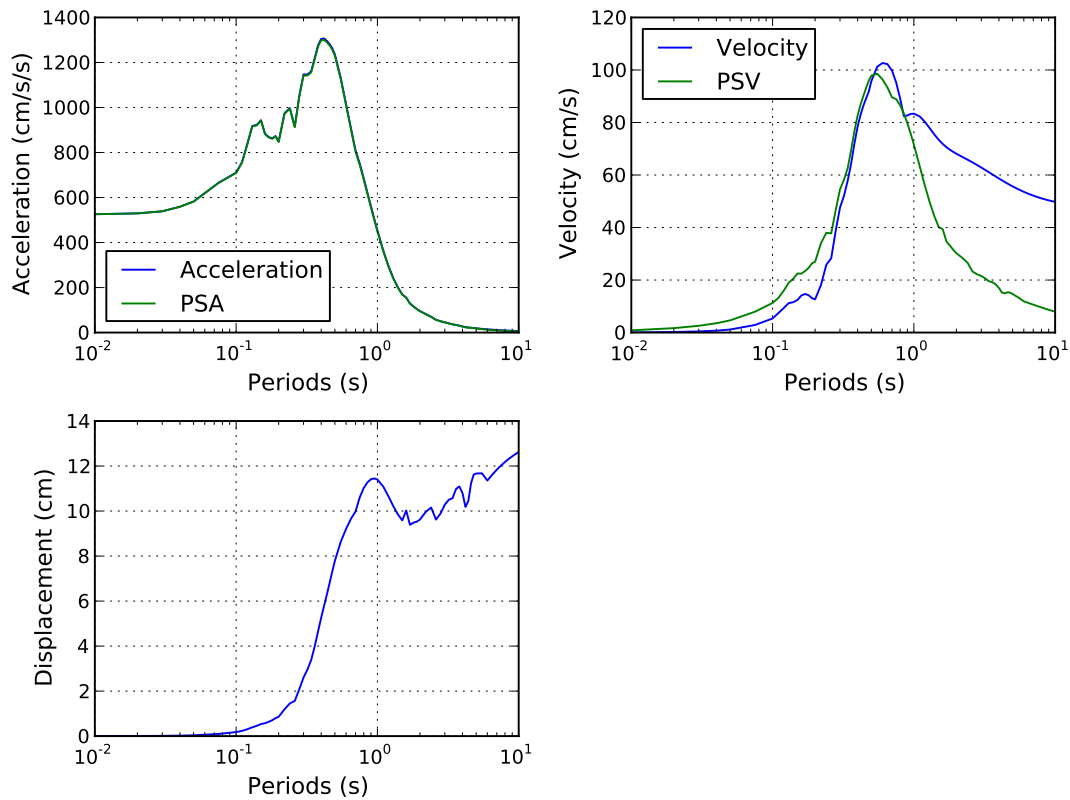


Figure 3.3 – Full response spectra of the waveform shown in Figure 3.1, calculated using the Nigam & Jennings (1969) method

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:

```
1 spectra_x, spectra_y = ims.get_response_spectrum_pair(
2     accel_x,
3     time_step, # Time-step of x-component
4     accel_y,
5     time_step, # Time-step of y-component,
6     periods,
7     damping=0.05,
8     units="cm/s/s",
9     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, $\text{spectra_x}(Sa_x(T, \xi))$ and $\text{spectra_y}(Sa_y(T, \xi))$, we can now obtain the following “resolved” horizontal spectra:

- **Geometric Mean**

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

Calculated using the following command:

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

- **Arithmetic Mean**

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

Calculated using the following command:

```
1 | 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:

```
1 | 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:

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

```
1 | 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 :

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

This can be achieved using the `ims` tools:

```
1 | rot_hist_x, rot_hist_y = ims.rotate_horizontal(accel_x,
2 |                                              accel_y,
3 |                                              theta)
```

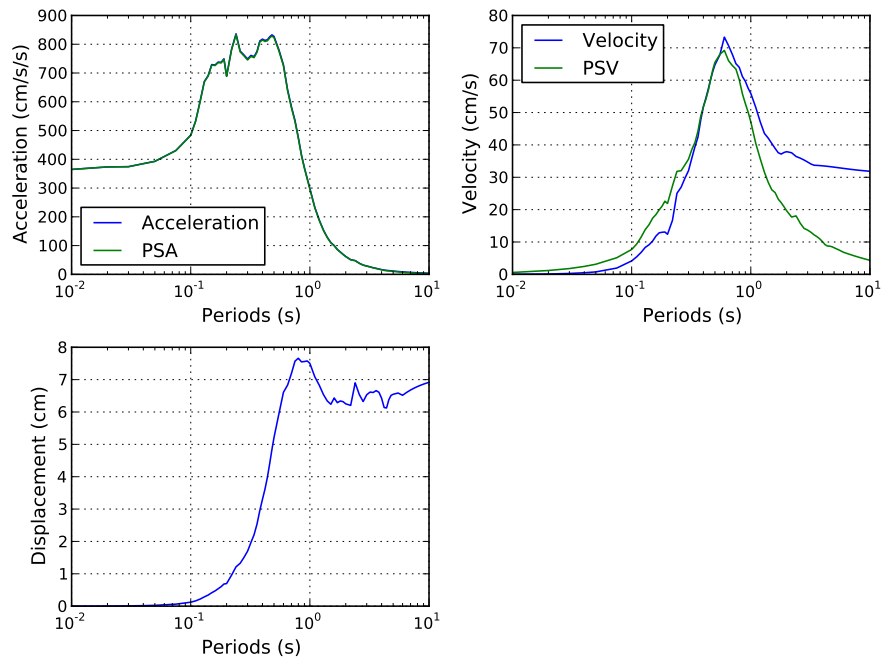


Figure 3.4 – Geometric mean spectra of the two horizontal records

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:

```

1 gmrotd50 = ims.gmrotdpp(accel_x, time_step,
2                           accel_y, time_step,
3                           periods,
4                           50.0, # Percentile
5                           damping=0.05,
6                           units="cm/s/s",
7                           method="Nigam-Jennings")
8 gmroti50 = ims.gmrotipp(accel_x, time_step,
9                           accel_y, time_step,
10                          periods,
11                          50.0, # Percentile
12                          damping=0.05,
13                          units="cm/s/s",
14                          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

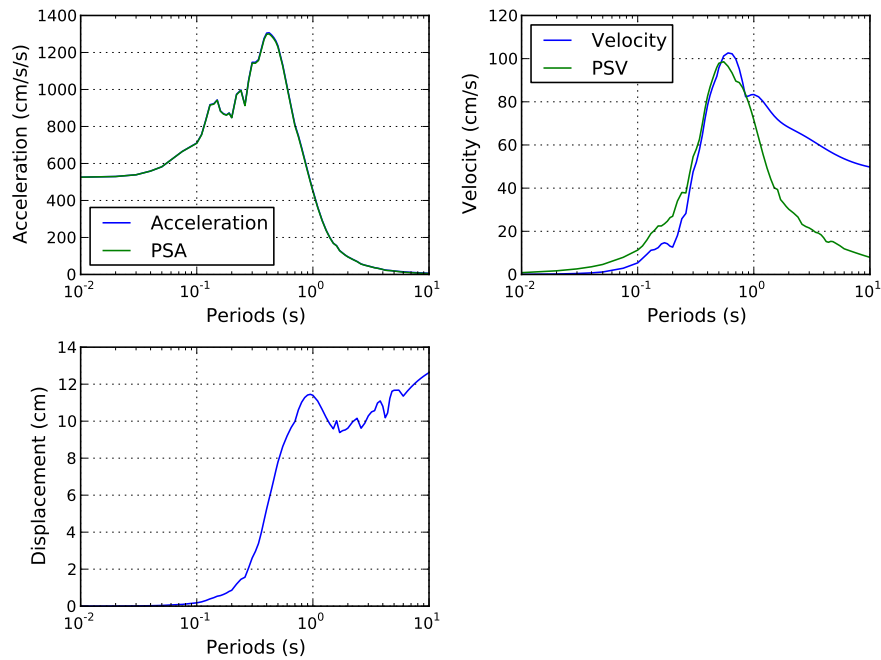


Figure 3.5 – Envelope spectra of the two horizontal records

constructed in the manner described in section ??, 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:

```

1 # Import the function add_horizontal_im
2 from smtk.sm_database_builder import add_horizontal_im
3 # Load in the database metadata file
4 import cPickle
5 database = cPickle.load(open("path/to/metadata.pkl", "r"))
6 # Choose the desired horizontal intensity measures
7 # The following will add the horizontal PGA, PGV
8 # Geometric mean spectrum, envelope spectrum and GMRotI50
9 im_list = ["PGA", "PGV", "Geometric", "Envelope", "GMRotI50"]
10 add_horizontal_im(database,
11                   im_list,
12                   component="Geometric",
13                   damping="05",
14                   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.

Trellis Plots: Advantages & Limitations

Trellis Plotting: The (Not So) Simple Way

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

Trellis Plotting: The Simple Way

- Configuring the Sites
- Creating the Trellis Points

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 used 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:

```
1 | 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:

```
1 | # Import the get_available_gsim function from OpenQuake
2 | from openquake.hazardlib.gsim import get_available_gsim
3 | # Show list of gsim
4 | get_available_gsim()
```

In the examples shown throughout this chapter we consider six GMPEs: `citeAkkarBommer2010`, `citeAkkarCagnan2010`, `siteAkkar2014` (Joyner-Boore coefficients), `citeBooreAtkinson2008`, `citeChiouYoungs2008`, `citeZhaoEtAl2006`. We also compare using just four intensity measures: PGA, $S_a(0.2s)$, $S_a(1.0s)$ and $S_a(2.0s)$.

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:

```
1 | gmpe_list = ["AkkarBommer2010",
2 |             "AkkarCagnan2010",
3 |             "AkkarEtAlRjb2014",
4 |             "BooreAtkinson2008",
5 |             "ChiouYoungs2008",
6 |             "ZhaoEtAl2006Asc"]
7 |
8 | imts = ["PGA", "SA(0.2)", "SA(1.0)", "SA(2.0)"]
9 | params = {"ztor": 5.0,      # Top of rupture depth
10 |          "hypo_depth": 10.0, # Hypocentral depth
11 |          "vs30": 800.0, # Vs30 for all sites
12 |          "vs30measured": True, # Vs30 value is measured
13 |          "z1pt0": 100.0, # Depth (m) to the 1.0 km/s Vs interface
14 |          "dip": 90.0, # Vertical Fault
15 |          "rake": 0.0 # Strike-slip fault
16 | }
```

The GMPEs and intensity measures must be input as a list, whilst missing parameters are 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:

```
1 # Import the numerical python tool
2 import numpy as np
3 # Generate magnitudes from 4.5 to 8.0 every 0.1
4 magnitudes = np.arange(4.5, 8.1, 0.1)
5 distances = {"repi": 20.0,
6             "rhypo": 22.5,
7             "rjb": 15.0,
8             "rrup": 16.0,
9             "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 will 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

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

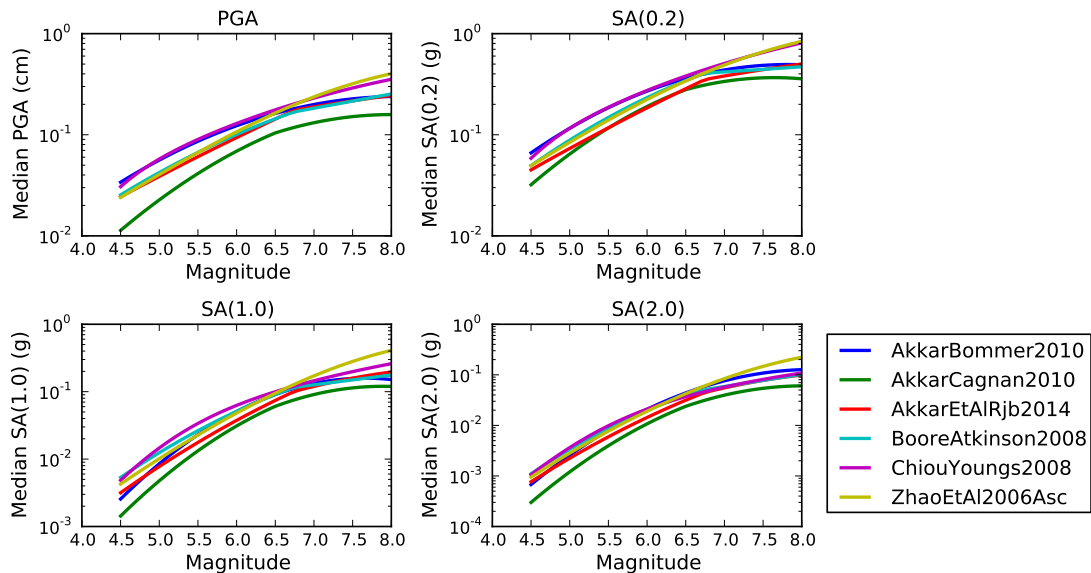


Figure 4.1 – Scaling of GMPEs with respect to magnitude

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:

```

1 trpl.MagnitudeSigmaIMTTrellis(magnitudes,
2                               distances,
3                               gmpe_list,
4                               imts,
5                               params,
6                               stddevs="Total",
7                               figure_size=(7,5),
8                               filename="/path/to/plot",
9                               filetype="png")

```

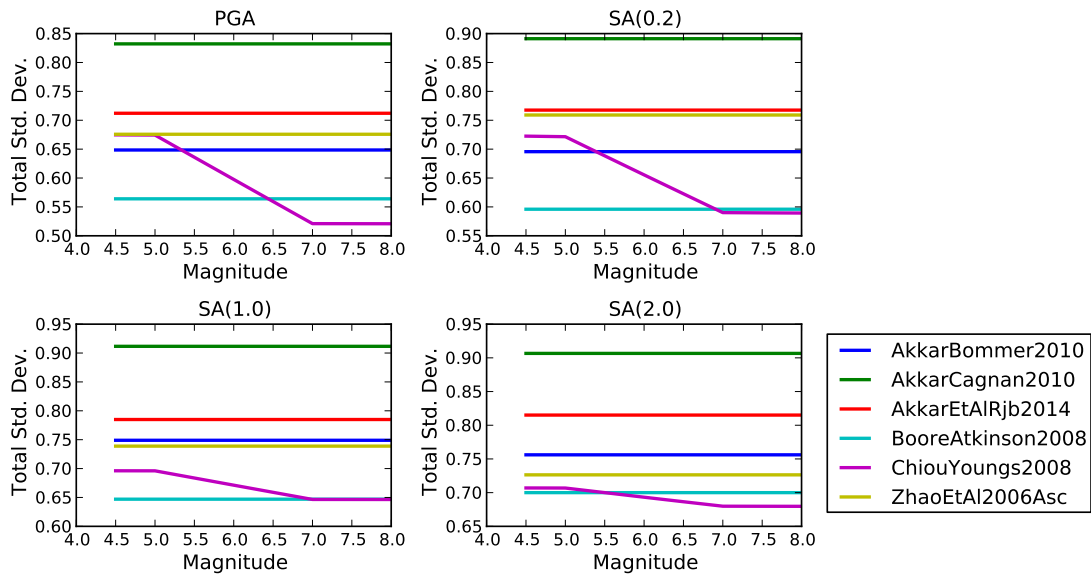


Figure 4.2 – Scaling of total standard deviation of the GMPEs with respect to magnitude

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:

```

1 magnitude = 6.5
2 distances = {"repi": np.arange(0.0, 151.0, 1.0)}
3 distances["rhypo"] = np.sqrt(distances["repi"] ** 2.0 +
4                               params["hypo_depth"] ** 2.0)
5 distances["rjb"] = distances["repi"]
6 distances["rrup"] = np.sqrt(distances["rjb"] ** 2.0 +
7                               params["ztor"] ** 2.0)
8 distances["rx"] = distances["rjb"]

```

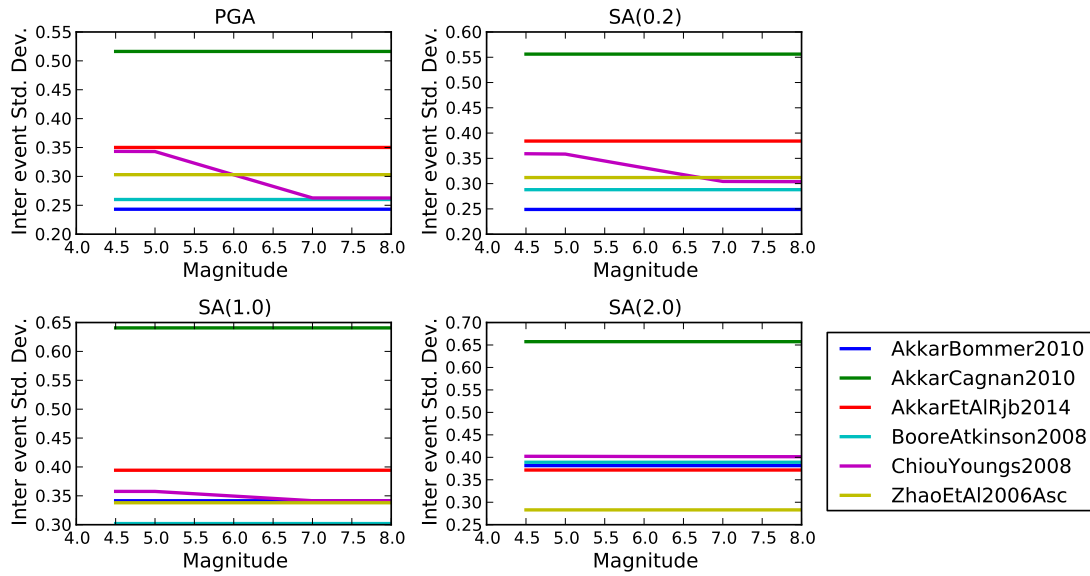


Figure 4.3 – Scaling of inter-event deviation of the GMPEs with respect to magnitude

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

```
1 trpl.DistanceIMTTrellis(magnitude,
2                           distances,
3                           gmpe_list,
4                           imts,
5                           params,
6                           distance_type="rjb",
7                           plot_type="loglog",
8                           filename="path/to/image",
9                           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 user 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:

```
1 trpl.DistanceSigmaIMTTrellis(magnitude,
2                               distances,
3                               gmpe_list,
4                               imts,
5                               params,
6                               stddevs="Total",
7                               distance_type="rjb",
8                               plot_type="loglog",
9                               filename="path/to/image",
10                              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

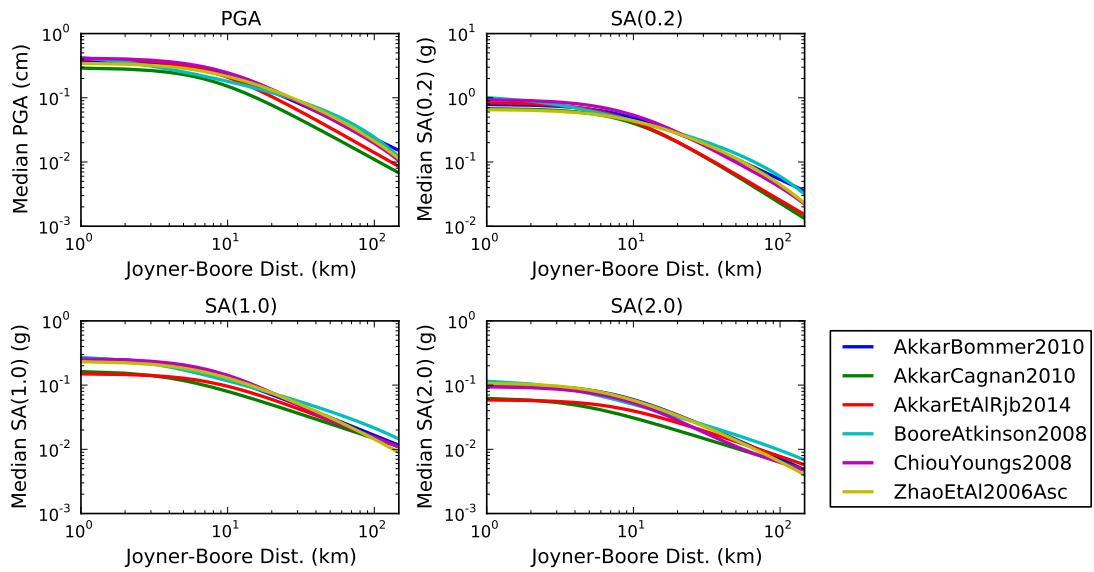


Figure 4.5 – Scaling of the GMPEs with respect to distance

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

```

1 | trpl.MagnitudeDistanceSpectraSigmaTrellis(magnitudes,
2 |                                             distances,
3 |                                             gmpe_list,
4 |                                             periods,
5 |                                             params,
6 |                                             plot_type="semilogy",
7 |                                             figure_size=(10,8),
8 |                                             filename="path/to/image",
9 |                                             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:

```

1 | 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

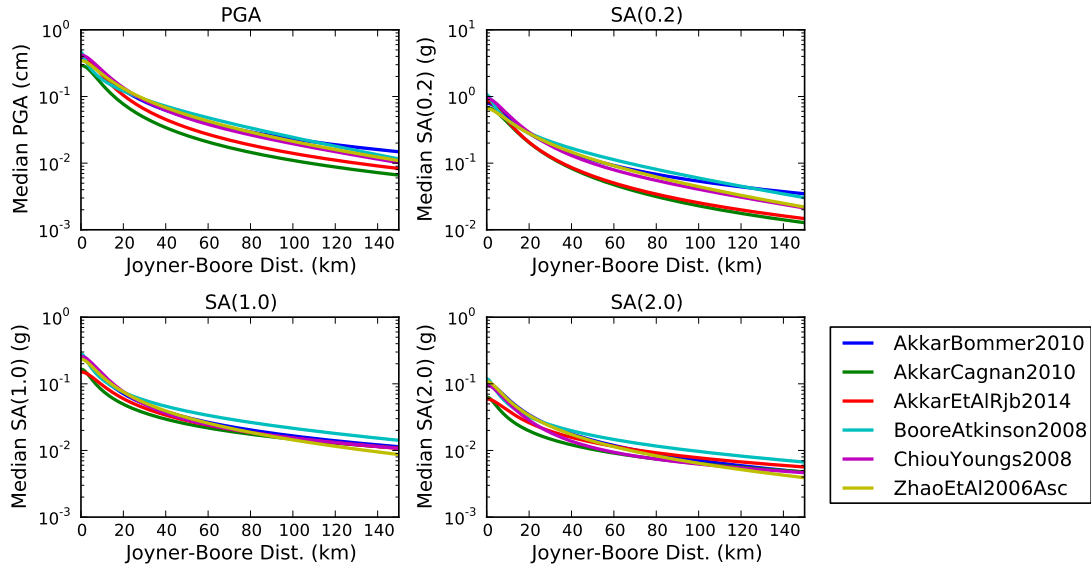


Figure 4.6 – As Figure 4.5 with linear distance axes

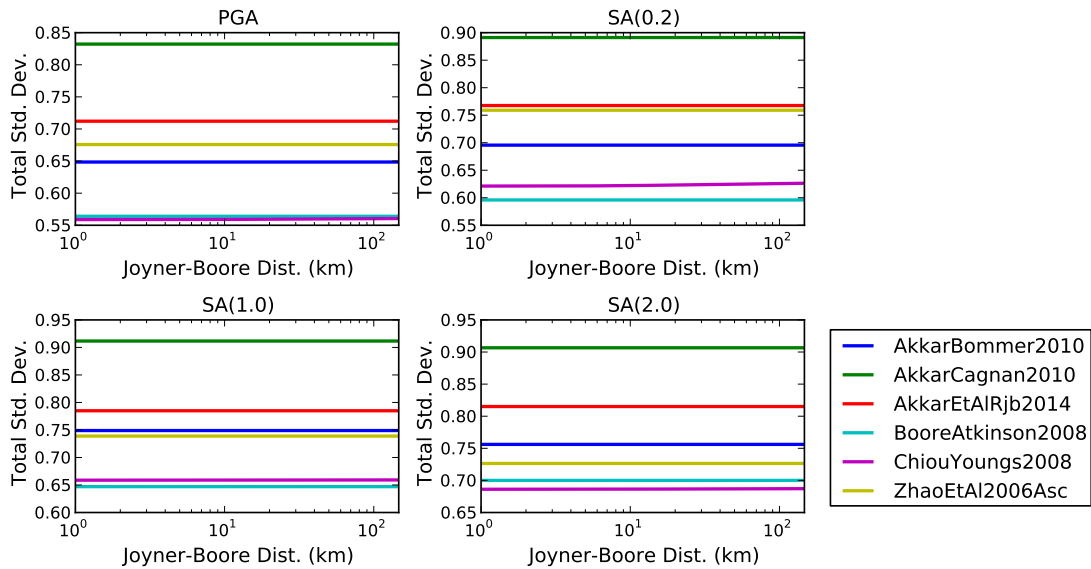


Figure 4.7 – Scaling of the total standard deviation with distance.

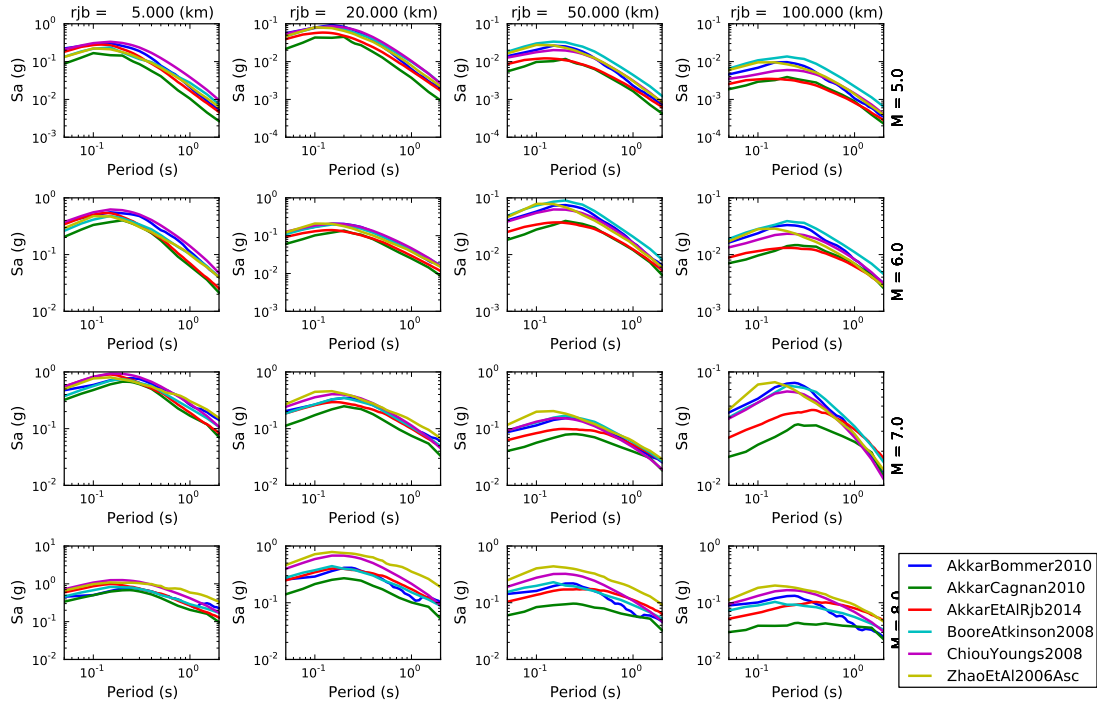


Figure 4.8 – Scaling of the response spectrum for each GMPE for selected magnitude and source-site distance combinations

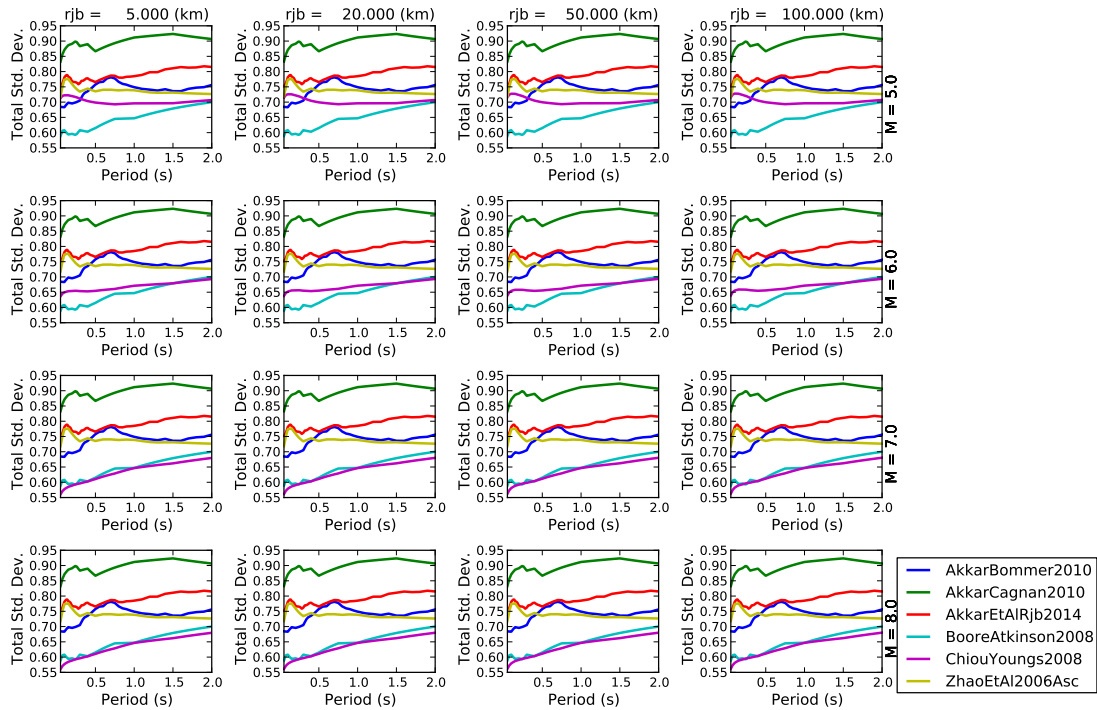


Figure 4.9 – Scaling of the response spectrum total standard deviation for each GMPE for selected magnitude and source-site distance combinations

- `msr` Magnitude scaling relation, default to **WellsCoppersmith1994**
- `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:

```
1 | rupt1 = rcfg.GSIMRupture(magnitude=6.5,
2 |                           dip=45.0,
3 |                           aspect=1.5,
4 |                           rake=90.0,
5 |                           ztor=0.0,
6 |                           strike=0.0,
7 |                           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`:

```
1 | sites_mesh = rupt1.get_target_sites_mesh(maximum_distance=200.0,
2 |                                           spacing=2.0,
3 |                                           vs30=800.0,
4 |                                           vs30measured=True,
5 |                                           z1pt0=None,
6 |                                           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:

```
1 | rupt1.plot_model_configuration(figure_size=(7,5),
2 |                               filename="/path/to/image",
3 |                               filetype="png",
4 |                               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


```

5 |                                     vs30measured=True ,
6 |                                     z1pt0=None ,
7 |                                     z2pt5=None )
8 | rupt1.plot_model_configuration()

```

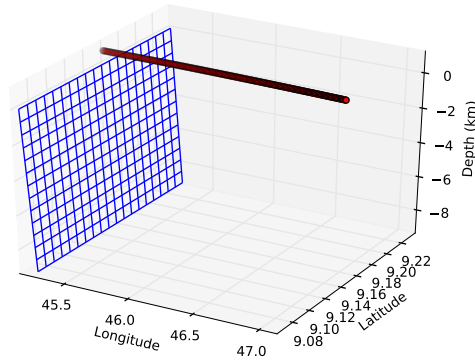


Figure 4.12 – Rupture and site configuration for a regularly spaced line or points propagating at an angle of 90° from the rupture, along the hanging wall

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:

```

1 | sites_line = rupt1.get_target_sites_point(distance=20.0 ,
2 |                                           distance_type="rjb" ,
3 |                                           vs30=800.0 ,
4 |                                           line_azimuth=90.
5 |                                           vs30measured=True ,
6 |                                           z1pt0=None ,
7 |                                           z2pt5=None )
8 | 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:

```

1 | trpl.DistanceIMTTrellis.from_rupture_model(
2 |     rupt1 ,
3 |     gmpe_list ,
4 |     imts ,
5 |     filename="path/to/image" ,
6 |     filetype="pdf")

```

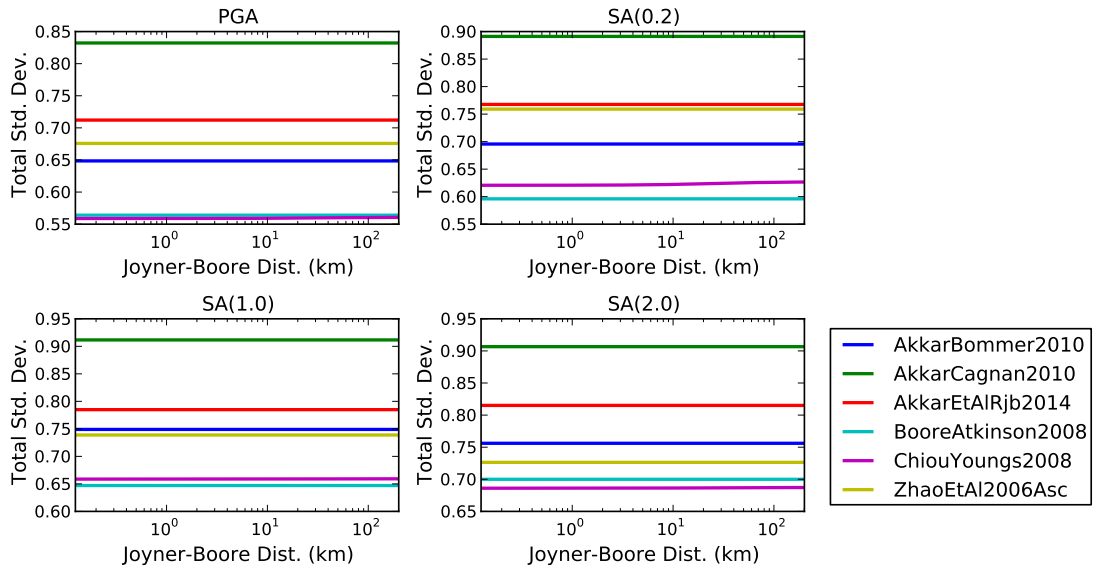



Figure 4.15 – Scaling of the total standard deviation GMPEs with respect to distance for the rupture and site configuration in Figure 4.12

```

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

```

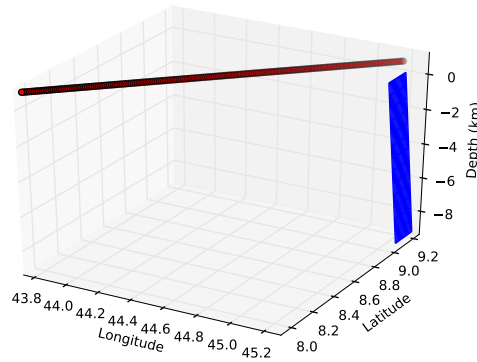


Figure 4.16 – Rupture and site configuration for a regularly spaced line or points propagating at an angle of 90° from the rupture, along the footwall

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:

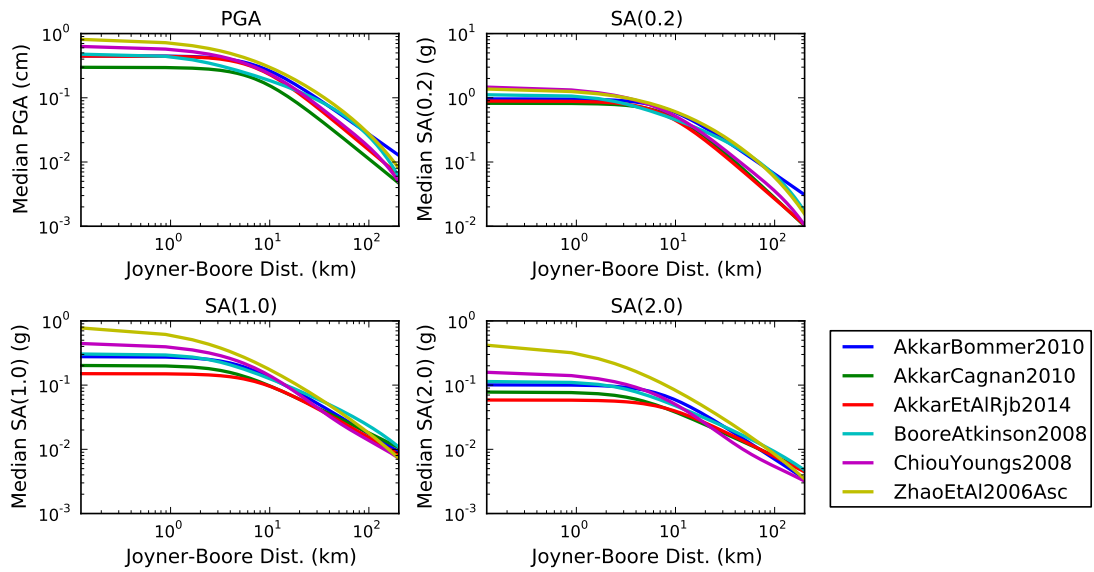


Figure 4.17 – Scaling of the GMPEs along the footwall of a rupture, configured according to Figure 4.16

```

1 sites_line = rupt1.get_target_sites_point(distance=20.0,
2                                           distance_type="rjb",
3                                           vs30=800.0,
4                                           line_azimuth=90.
5                                           vs30measured=True,
6                                           z1pt0=None,
7                                           z2pt5=None)
8 # Create the expected ground motion trellis plot
9 trpl.MagnitudeDistanceSpectraTrellis.from_rupture_model(
10     rupt1,
11     gmpe_list,
12     periods,
13     filename="path/to/image",
14     filetype="png")
15 trpl.MagnitudeDistanceSpectraSigmaTrellis.from_rupture_model(
16     rupt1,
17     gmpe_list,
18     periods,
19     stddevs="Total",
20     filename="path/to/image",
21     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:

```

1 sites_line = rupt1.get_target_sites_point(distance=20.0,
2                                           distance_type="rjb",
3                                           vs30=800.0,
4                                           line_azimuth=220.
5                                           vs30measured=True,
6                                           z1pt0=None,
7                                           z2pt5=None)
8 rupt1.plot_model_configuration()

```

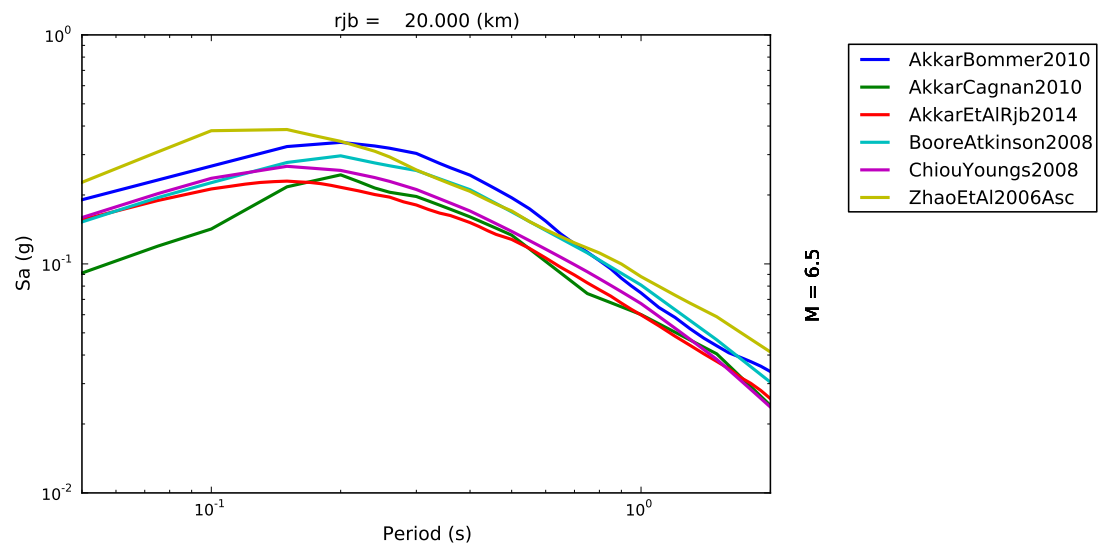


Figure 4.18 – Comparison of response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.13

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

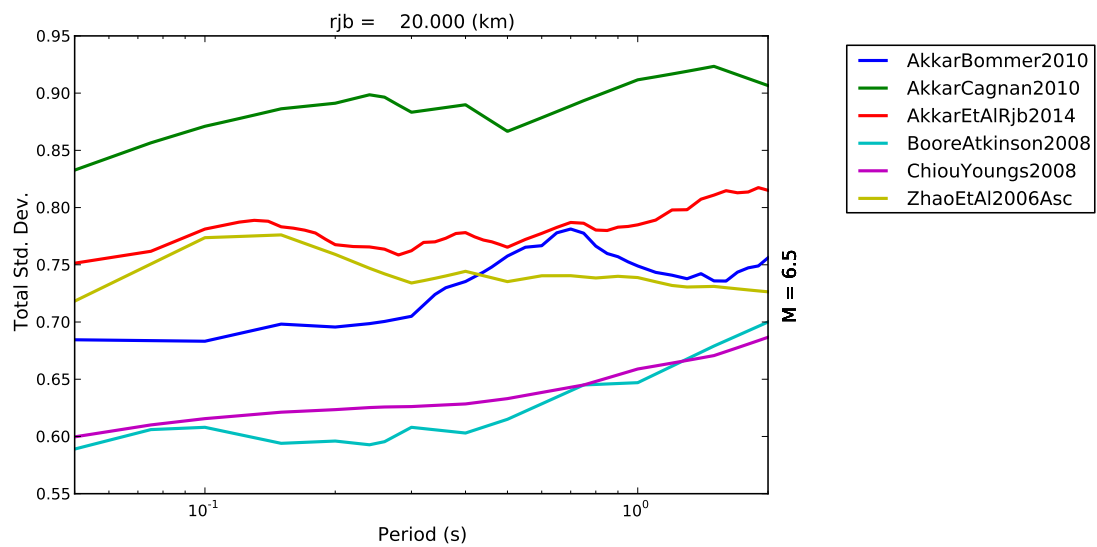


Figure 4.19 – Comparison of total standard deviation of the response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.13

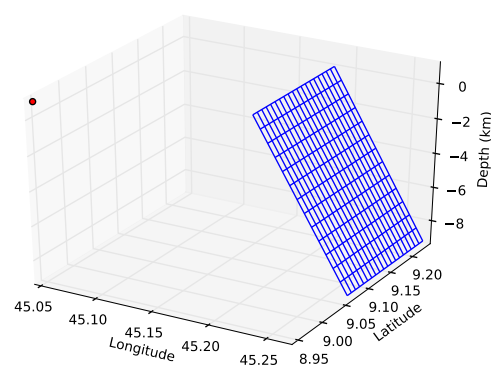


Figure 4.20 – Rupture and site configuration for a single point located at 20 km (Joyner-Boore) from the rupture on a bearing of 220° on the hanging wall

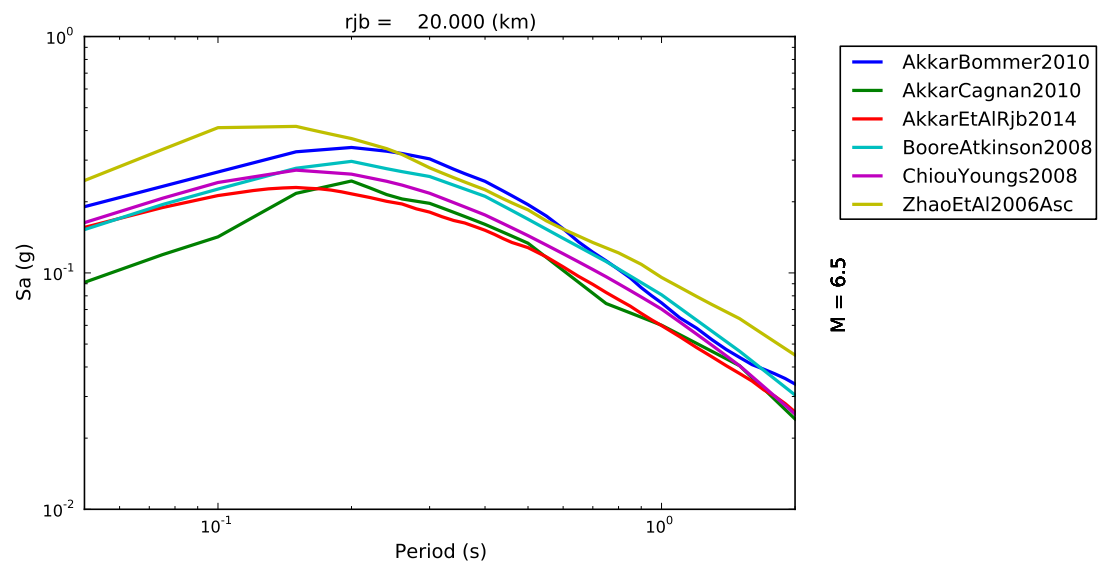


Figure 4.21 – Comparison of response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.20

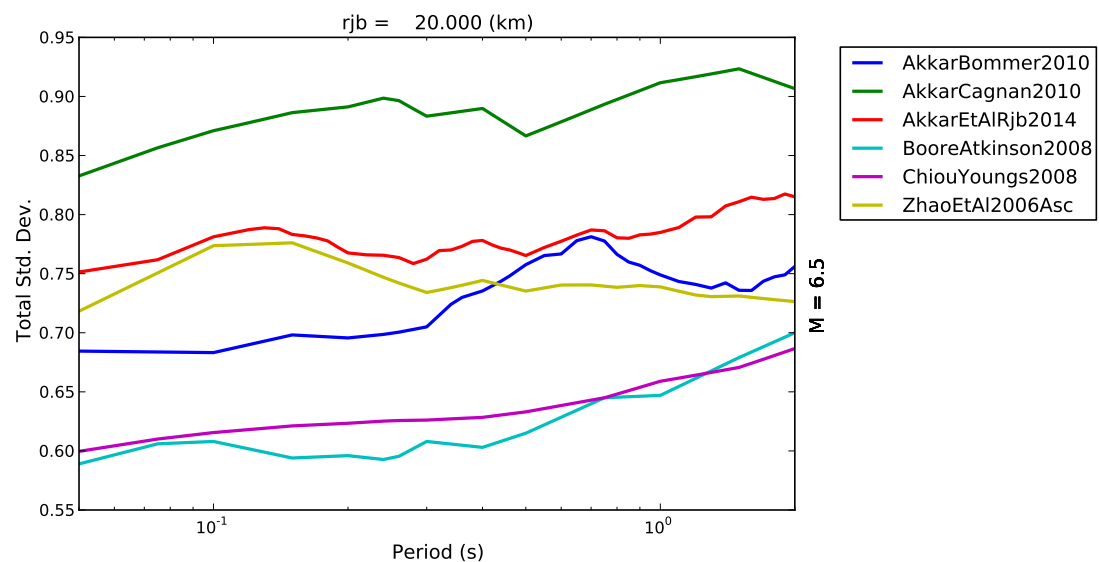


Figure 4.22 – Comparison of total standard deviation of the response spectra from the GMPEs for the rupture and site configuration shown in Figure 4.20

GMPE Residuals

Definition of the GMPE Residuals
Residual analysis in the GMPE-SMTK

Basic Trends in Residuals

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

Residual Trends with Predictor Variables

Log-likelihood Approach

Euclidean Distance Based Ranking

Site Specific Analysis

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 represent the local properties of source, path and site scaling of the ground motion. In modelling both the expected ground motion and the aleatory variability, each GMPE is a probability distribution in which a ground motion y_{ij} at recorded location j , originating from event i , is characterised by a lognormal distribution:

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

where $\mu(m_i, r_{ij}, \theta_{ij})$ is the expected ground motion from an event of magnitude m_i , recorded at a distance r_{ij} , where θ_{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:

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

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:

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

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:

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

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:

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

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

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

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
```

```

3 # Import the database
4 sm_database = cPickle.loads(open("path/to/metadata.pkl", "r"))
5 # Define the list of GMPEs
6 gmpe_list = ["BooreAtkinson2008",
7              "AkkarBommer2010",
8              "AkkarCagnan2010",
9              "AkkarEtAlRjb2014"]
10 # Define the list of intensity measures
11 imt_list = ["PGA", "SA(0.2)", "SA(1.0)", "SA(2.0)"]

```

For undertaking simple analysis of residuals simply run:

```

1 # Set-up the residual calculator
2 resid1 = res.Residuals(gmpe_list, imts)
3 # Run the residual calculator
4 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:

```

1 | 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_{Aij} 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 $S_a(1.0s)$ using Akkar et al (2014) only in Figure 5.2.

```

1 # Boore & Atkinson (2008)
2 rspl.ResidualPlot(resid1, "BooreAtkinson2008", "PGA",
3                   filename="path/to/image.png",
4                   filetype="png")
5 # Akkar & Bommer (2010)
6 rspl.ResidualPlot(resid1, "AkkarBommer2010", "PGA")
7 # Akkar & Cagnan (2010)
8 rspl.ResidualPlot(resid1, "AkkarCagnan2010", "PGA")
9 # Akker et al. (2014)
10 rspl.ResidualPlot(resid1, "AkkarEtAlRjb2014", "PGA")
11 # Akker et al. (2014) - Sa (1.0 s)
12 rspl.ResidualPlot(resid1, "AkkarEtAlRjb2014", "Sa(1.0)")

```

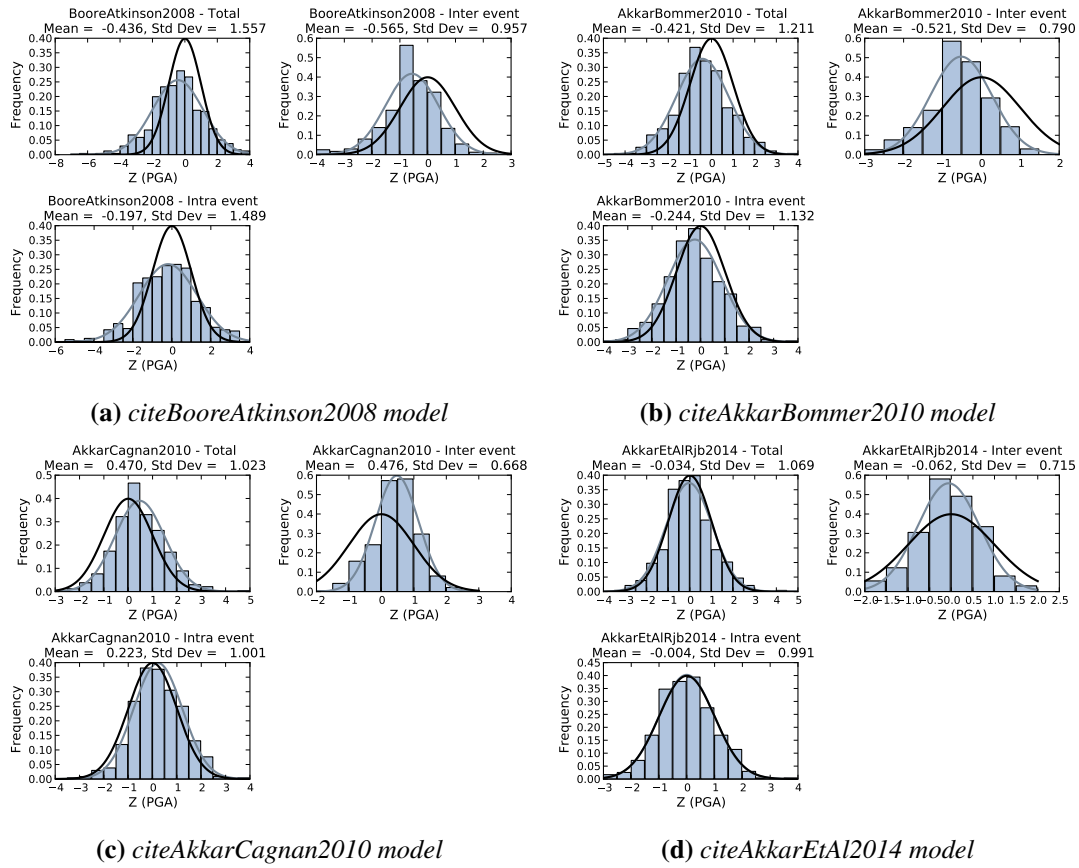


Figure 5.1 – PGA residual distribution for the record set. The grey line indicates the probability density function fit to the data, whilst the black line indicates the standard normal density function

Figures 5.1 and 5.2 shows a relatively discernible set of trends. Consider just the **boore2008** 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 **boore2008** model 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 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 $S_a(1.0s)$.

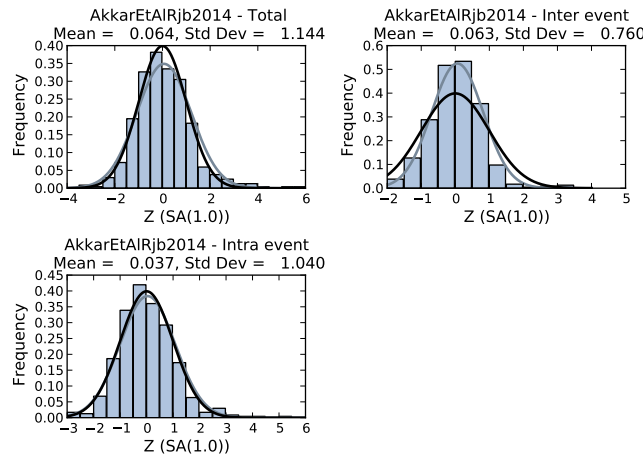


Figure 5.2 – PGA residual distribution for the record set using the Akkar et al. (2014) GMPE

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 be presented later in the chapter) is to consider the overall goodness-of-fit of the model to the predictions. SCHERBAUM ET AL 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.

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

The total “likelihood” of a particular value is given by:

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

As indicated in SCHERBAUM ET AL 2004 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.Likelihood` tool operates in the same manner as the `res.Residuals` tool; therefore to run the likelihood analysis one follows the same approach:

```
1 | # Set-up the likelihood calculator
2 | lh1 = res.Likelihood(gmpe_list, imts)
3 | # Run the residual calculator
4 | 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:

```

1 # Boore & Atkinson (2008)
2 rspl.LikelihoodPlot(lh1, "BooreAtkinson2008", "PGA",
3                       filename="path/to/image.png",
4                       filetype="png")
5 # Akkar & Bommer (2010)
6 rspl.LikelihoodPlot(lh1, "AkkarBommer2010", "PGA")
7 # Akkar & Cagnan (2010)
8 rspl.LikelihoodPlot(lh1, "AkkarCagnan2010", "PGA")
9 # Akker et al. (2014)
10 rspl.LikelihoodPlot(lh1, "AkkarEtAlRjb2014", "PGA")
11 # Akker et al. (2014) - Sa (1.0 s)
12 rspl.LikelihoodPlot(lh1, "AkkarEtAlRjb2014", "Sa(1.0)")

```

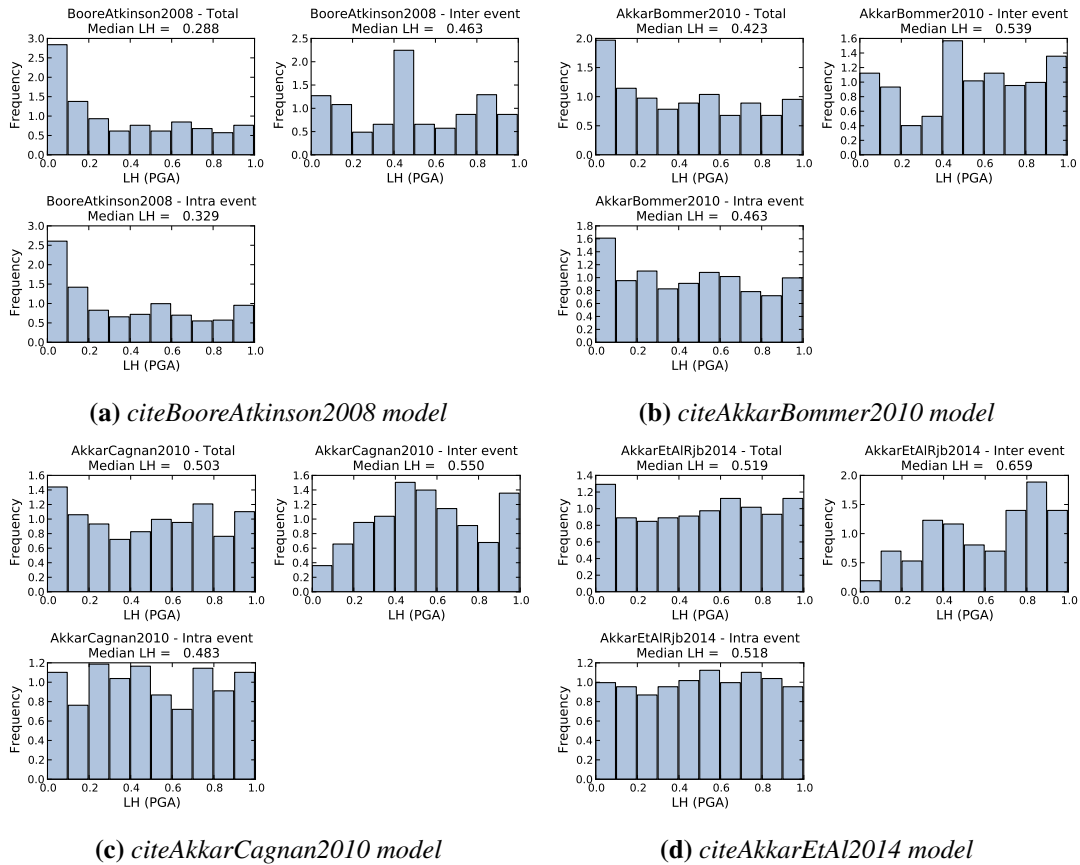


Figure 5.3 – LH distribution for the observed PGA set

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 **boore2008** 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).

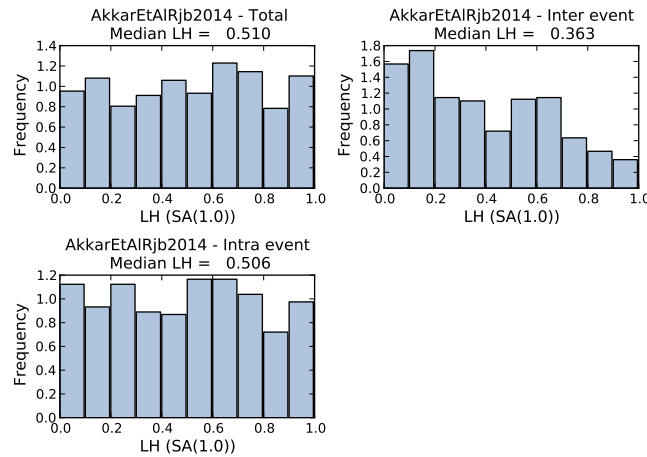


Figure 5.4 – LH distribution for the observed $S_a(1.0s)$ set using the Akkar et al. (2014) GMPE

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 **boore2008** and Akkar et al. (2014) GMPEs for PGA and $S_a(1.0s)$. These figures were generated with the following commands:

```

1 # Boore & Atkinson (2008) - PGA
2 rspl.ResidualWithMagnitude(resid1, # Residuals
3                             "BooreAtkinson2008", # GMPE
4                             "PGA", # Intensity Measure
5                             filename="path/to/image.png",
6                             filetype="png")
7 # Akkar et al. (2014) - PGA
8 rspl.ResidualWithMagnitude(resid1, "AkkarEtAlRjb2014", "PGA")
9 # Boore & Atkinson (2008) - Sa (1.0)
10 rspl.ResidualWithMagnitude(resid1, "BooreAtkinson2008",
11                             "Sa(1.0)")
12 # Akkar et al. (2014) - Sa (1.0)
13 rspl.ResidualWithMagnitude(resid1, "AkkarEtAlRjb2014",
14                             "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 largest magnitudes, whilst for $S_a(1.0s)$ the opposite is true, indicating the GMPE is predicting lower than expected ground motions at higher magnitudes.

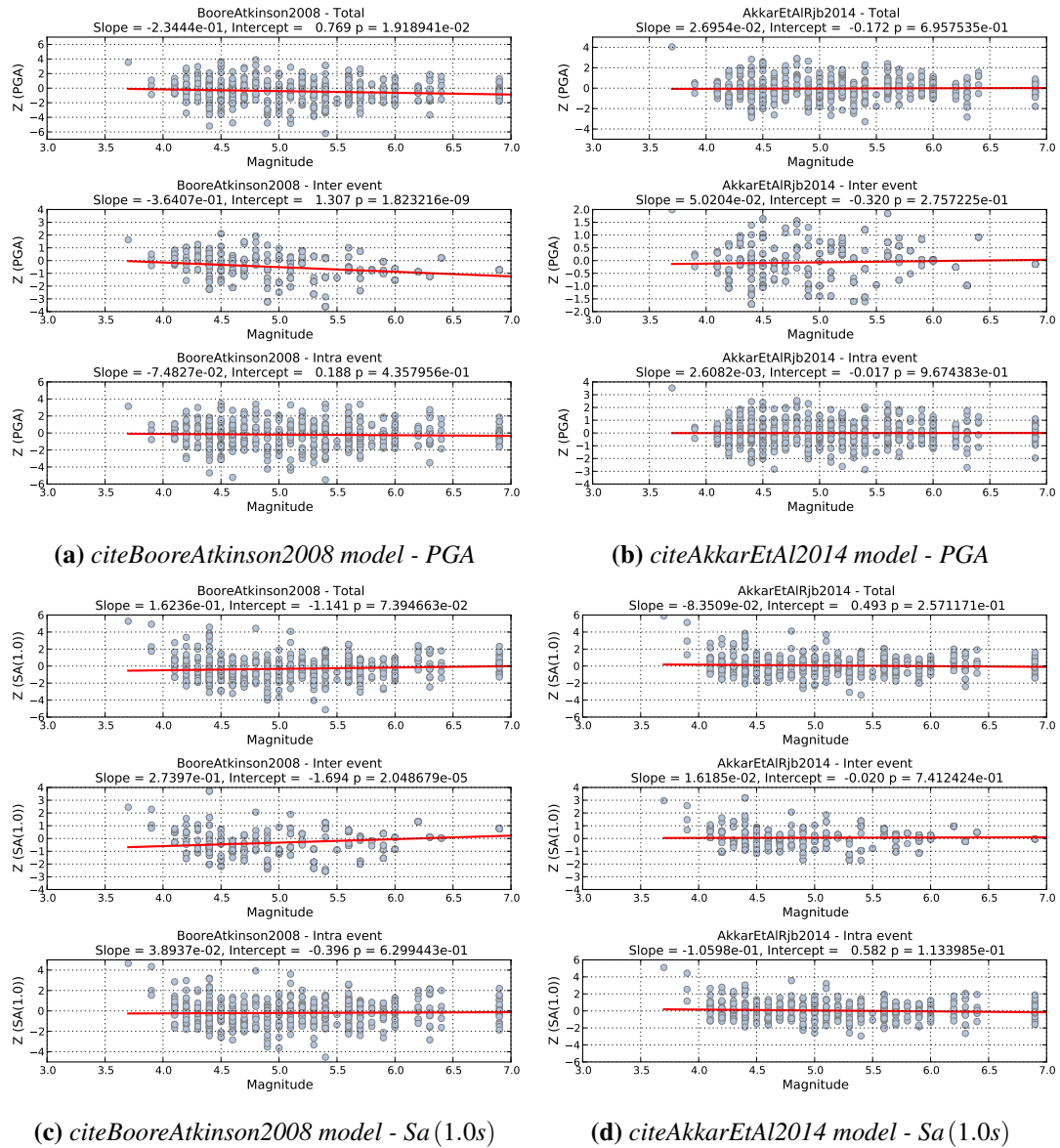


Figure 5.5 – Residual trends with magnitude for the observed PGA and $S_a(1.0s)$ records

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:

```

1 # Boore & Atkinson (2008) - PGA
2 rspl.ResidualWithDistance(resid1, # Residuals
3                             "BooreAtkinson2008", # GMPE
4                             "PGA", # Intensity Measure
5                             distance_type="rjb"
6                             plot_type="linear",
7                             filename="path/to/image.png",
8                             filetype="png")
9 # Akkar et al. (2014) - PGA
10 rspl.ResidualWithDistance(resid1, "AkkarEtAlRjb2014",
11                             "PGA", distance_type="rjb",
12                             plot_type="linear")
13 # Boore & Atkinson (2008) - Sa (1.0)
14 rspl.ResidualWithDistance(resid1, "BooreAtkinson2008",
15                             "Sa(1.0)", distance_type="rjb",
16                             plot_type="linear")
17 # Akkar et al. (2014) - Sa (1.0)
18 rspl.ResidualWithDistance(resid1, "AkkarEtAlRjb2014",
19                             "Sa(1.0)", distance_type="rjb",
20                             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:

```

1 # Boore & Atkinson (2008) - PGA
2 rspl.ResidualWithVs30(resid1, # Residuals
3                         "BooreAtkinson2008", # GMPE
4                         "PGA", # Intensity Measure
5                         filename="path/to/image.png",
6                         filetype="png")
7 # Akkar et al. (2014) - PGA
8 rspl.ResidualWithVs30(resid1, "AkkarEtAlRjb2014", "PGA")
9 # Boore & Atkinson (2008) - Sa (1.0)
10 rspl.ResidualWithVs30(resid1, "BooreAtkinson2008", "Sa(1.0)")

```

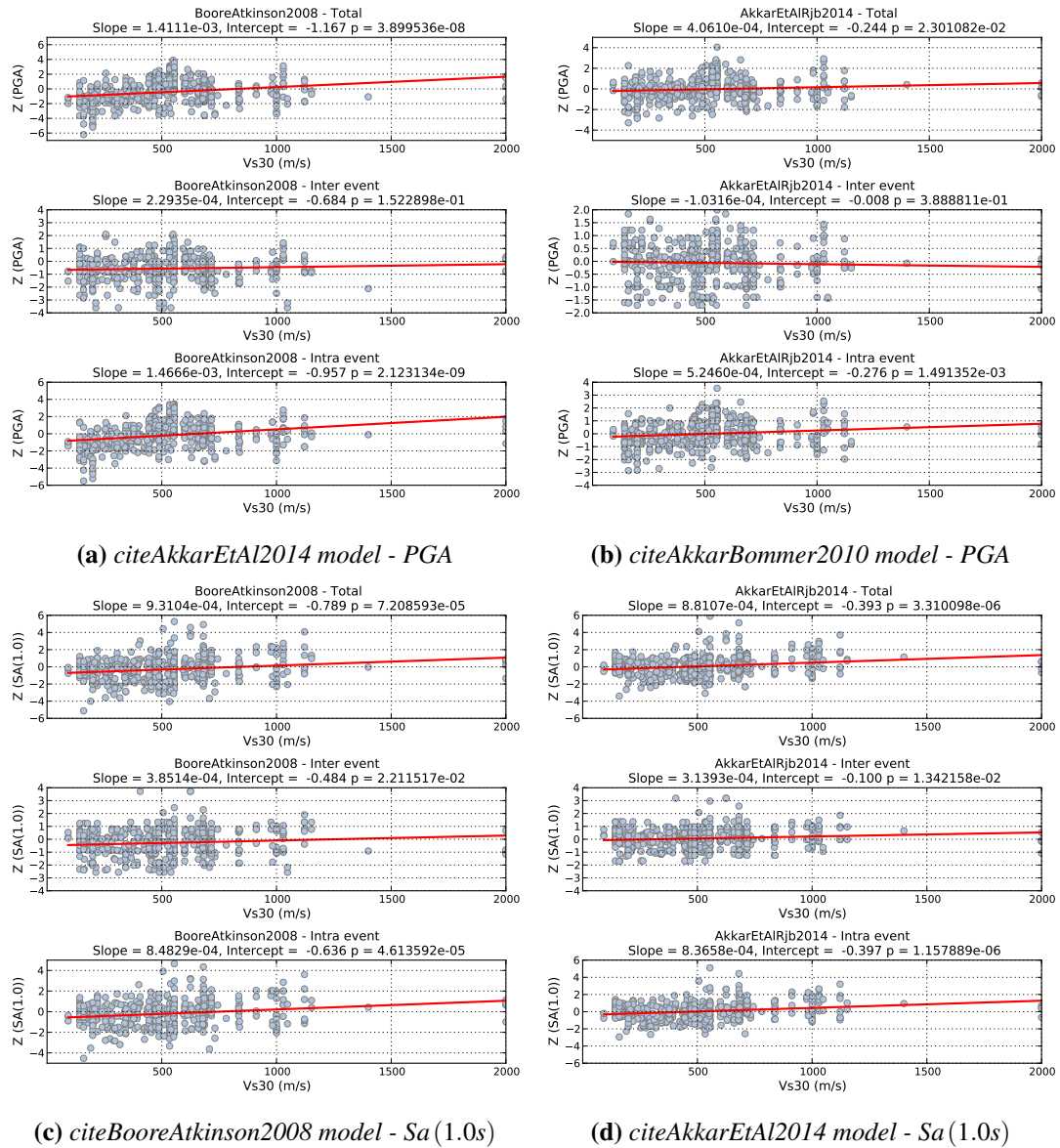



Figure 5.7 – Residual trends with V_{S30} for the observed PGA and Sa(1.0s) records

```

4 |                                     "PGA",    # Intensity Measure
5 |                                     filename="path/to/image.png",
6 |                                     filetype="png")
7 | # Akkar et al. (2014) - PGA
8 | rspl.ResidualWithDepth(resid1, "AkkarEtAlRjb2014", "PGA")
9 | # Boore & Atkinson (2008) - Sa (1.0)
10 | rspl.ResidualWithDepth(resid1, "BooreAtkinson2008", "Sa(1.0)")
11 | # Akkar et al. (2014) - Sa (1.0)
12 | 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.

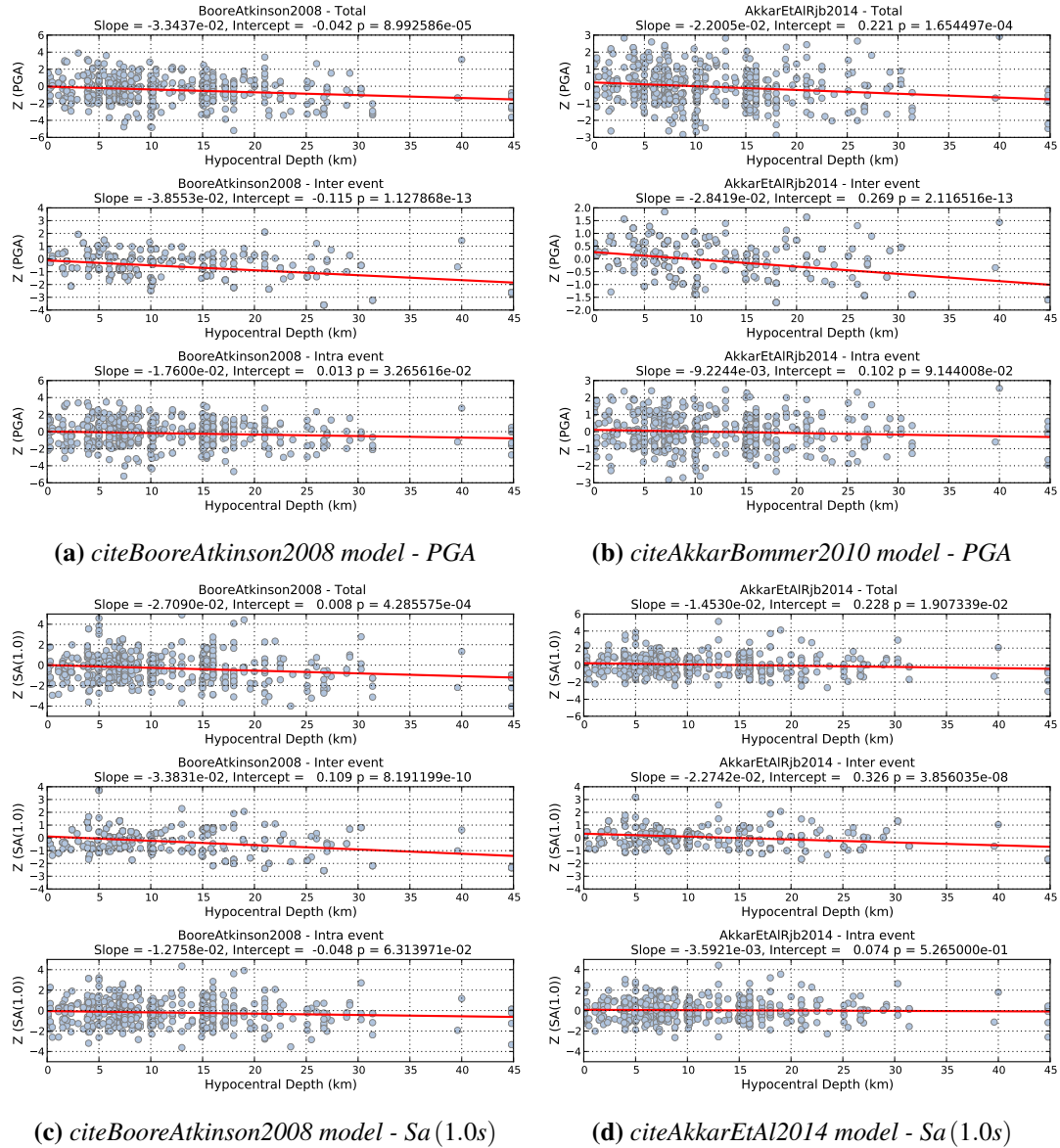


Figure 5.8 – Residual trends with hypocentral depth for the observed PGA and Sa (1.0s) records

5.5 Log-likelihood Approach

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

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

```
1 for gmpe in gmpe_list:
2     print "LLH(%s) = %12.6f Weight = %12.6f" %(gmpe,
3         llh_values[gmpe],
4         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

```
1 | # EDR analysis
2 | edr1 = res.EDR(gmpe_list, imt_list)
3 | edr1.get_residuals(italy_db_clean)
4 | edr_values = edr1.get_edr_values()
```

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

```
1 | for gmpe in gmpe_list:
2 |     print "EDR(%s) = %8.4f (sqrt(kappa) = %8.4f)" %(gmpe,
3 |         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

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



Bibliography

Books

Articles

Other Sources

